



AFRL-RH-WP-TR-2014-0052

Development and Utility of Automatic Language Processing Technologies, Volume II

Brian Ore

Jeremy Gwinnup

Stephen Thorn

Michael Hutt

David Hoeferlin

William Coate

David Snyder

SRA International

5000 Springfield Street, Suite 200

Dayton, OH 45431

Katherine Young

N-Space Analysis, LLC

305 Winding Trail

Xenia, OH 45385

APRIL 2014

Final Technical Report

Distribution A. Approved for public release; distribution is unlimited.

**AIR FORCE RESEARCH LABORATORY
711TH HUMAN PERFORMANCE WING
HUMAN EFFECTIVENESS DIRECTORATE
HUMAN-CENTERED ISR DIVISION
HUMAN TRUST AND INTERACTION BRANCH
WRIGHT-PATTERSON AFB OH 45433**

STINFO Copy

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report was cleared for public release by the 88th Air Base Wing Public Affairs Office and is available to the general public, including foreign nationals. Copies may be obtained from the Defense Technical Information Center (DTIC) (<http://www.dtic.mil>).

AFRL-RH-WP-TR-2014-0052 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

//SIGNED//

TIMOTHY R. ANDERSON, Ph.D.
Work Unit Manager
Human Trust and Interaction Branch

//SIGNED//

LOUISE A. CARTER, Ph.D.
Chief, Human-Centered ISR Division
Human Effectiveness Directorate
711th Human Performance Wing
Air Force Research Laboratory

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE				Form Approved OMB No. 0704-0188	
<p>Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Washington Headquarters Service, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington, DC 20503.</p> <p>PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.</p>					
1. REPORT DATE (DD-MM-YYYY) 01-04-2014		2. REPORT TYPE Final		3. DATES COVERED (From - To) 8 October 2008 – 31 March 2014	
4. TITLE AND SUBTITLE Development and Utility of Automatic Language Processing Technologies, Volume II				5a. CONTRACT NUMBER FA8650-09-D-6939-001	
				5b. GRANT NUMBER N/A	
				5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S) *Brian Ore *Jeremy Gwinnup *Stephen Thorn *Michael Hutt *David Hoeferlin *William Coate *David Snyder				5d. PROJECT NUMBER	
				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER H05R (2830X03W)	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) *SRA International, Inc. 5000 Springfield Street, Suite 200 Dayton, OH 45431				8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Materiel Command Air Force Research Laboratory 711th Human Performance Wing Human Effectiveness Directorate Human-Centered ISR Division Human Trust and Interaction Branch Wright-Patterson AFB, OH 45433				10. SPONSOR/MONITOR'S ACRONYM(S) 711 HPW/RHXS	
				11. SPONSORING/MONITORING AGENCY REPORT NUMBER AFRL-RH-WP-TR-2014-0052	
12. DISTRIBUTION AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.					
13. SUPPLEMENTARY NOTES 88ABW-2014-2558; Cleared 28 May 2014					
14. ABSTRACT This final report provides research results in the development and utility of automatic speech recognition (ASR), machine translation (MT), natural language processing (NLP), speech synthesis (TTS) and other speech and language processing technologies.					
15. SUBJECT TERMS Automatic speech recognition (ASR), machine translation (MT), natural language processing (NLP), and speech synthesis (TTS).					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT SAR	18. NUMBER OF PAGES 111	19a. NAME OF RESPONSIBLE PERSON Timothy R. Anderson
a. REPORT U	b. ABSTRACT U	c. THIS PAGE U			19b. TELEPHONE NUMBER (Include area code) N/A

THIS PAGE LEFT INTENTIONALLY BLANK.

TABLE OF CONTENTS

<u>Section</u>	<u>Page</u>
List of Figures	v
List of Tables	v
SUMMARY	1
1.0 INTRODUCTION	2
2.0 EXPERIMENTS AND ACCOMPLISHMENTS	3
2.1 MT	3
2.1.1. Champollion Extensions	3
2.1.2. MT09 Evaluation	3
2.1.3. IWSLT13 Evaluation	4
2.1.4. Qahira	5
2.1.5. Lader	5
2.1.6. LM Interpolation	5
2.1.7. SMT2 Improvements	5
2.1.8. Language Resource Identification	6
2.1.9. Character-Based Processing	7
2.1.10. Alignment	14
2.1.11. Improvements to the Urdu Lexicon	18
2.1.12. Morphological Processing	20
2.1.13. French-to-English MT Output	24
2.1.14. Transliteration	26
2.1.15. OOV Word Handling	29
2.1.16. Chinese Word Segmentation	38
2.1.17. Paraphrasing	40
2.1.18. Reverse Palladius (RevP) Project	45
2.1.19. Pashto-French-English Translation Project	45
2.1.20. Review of Previous Work and Available Tools for Chinese MT	46
2.1.21. SRILM	46
2.1.22. Trigger-Based Lexicon Model	47
2.1.23. Recurrent Neural Network LM (RNNLM)	47
2.1.24. Neural Probabilistic LM (NPLM)	48
2.1.25. Translation Web Services	48
2.1.26. Evaluation and Scoring	51
2.1.27. Summary Figure	51
2.1.28. Recommendations for Future Work	55
2.2 ASR	55
2.2.1. Sphinx-4 Models	56

2.2.2. Spoken Language Communication and Translation System for Tactical Use (TRANSTAC) Dari and Pashto	56
2.2.3. CALLHOME Mandarin.....	57
2.2.4. Fisher and WSJ English.....	58
2.2.5. Summary.....	59
2.2.6. Recommendations for Future Work.....	59
2.3 Speech Synthesis.....	59
2.3.1. TTS GUI	60
2.3.2. Audio Recorder GUI.....	60
2.2.3. Phone Alignment Tool.....	60
2.3.4. English Voices	61
2.3.5. Iraqi Voices.....	63
2.3.6. Mandarin Voices.....	64
2.3.7. Spanish Voices.....	64
2.3.8. Automation of Speaker Adaptation	64
2.3.9. STRAIGHT Evaluation	65
2.3.10. Summary	65
2.3.11. Recommendations for Future Work.....	66
2.4 Laboratory Corpora Support.....	66
2.4.1. Sada-e-Azadi Corpus	66
2.4.2. JRC Aquis Corpus	67
2.4.3. SETimes Corpus	68
2.4.4. Summary.....	68
2.4.5. Recommendations for Future Work.....	68
2.5 Laboratory System Administration Support.....	68
2.5.1. Ethernet Infrastructure Upgrade	69
3.0 CONCLUSIONS AND RECOMMENDATIONS	70
4.0 REFERENCES	72
5.0 LIST OF ACRONYMS & GLOSSARY	74
APPENDIX A – List of Russian MT Resources	79
APPENDIX B – Consonant-Qualified Transliteration.....	93

LIST OF FIGURES

<u>Figure</u>	<u>Page</u>
1	A Portion of the CyberTrans Interface for Translation..... 48
2	Example of Drop-Down Selection of Language, Dictionary and Translation Engine 50
3	Basic, Single-Line Phrase Interface Using XML-RPC with Moses Server..... 51
4	Directory Archival Tool..... 53
5	Interface from IBLEU Scoring a Specific Segment of Text 54
6	Text-to-Speech GUI..... 62
7	Audio Recorder GUI..... 62

LIST OF TABLES

<u>Table</u>	<u>Page</u>
1	WERs obtained on TRANSTAC Dari and Pashto..... 57
2	Methods for Applying Discriminative Training to the Callhome Mandarin HMM SRS. 58

SUMMARY

This document provides a summary of work completed by SRA International for the Human Language Technology (HLT) Group of 711 HPW/RHXS during the period 08 October 2008 to 31 March 2014 under the Information Operations Cyber Exploitation Research (ICER) contract FA8650-09-D-6939, Task Order 0001.

Air Force and Department of Defense (DoD) personnel are called upon to operate all over the world with a number of multinational operations including the Global War on Terror, humanitarian relief operations, various coalition operations, and foreign internal defense. With its global reach and responsibilities, the DoD needs to monitor and understand ongoing situations, anticipate new situations that will require responses, and where possible influence the outcomes. Much of the information needed to effectively understand these situations and to operate in them is found in foreign language speech and text; however, there is a critical lack of linguists to understand and translate this material. To address the linguist shortfall, the HLT group of 711 HPW/RHXS is investigating the development and utility of Automatic Speech Recognition (ASR), Machine Translation (MT), Natural Language Processing (NLP), Speech Synthesis (TTS) and other speech and language processing technologies.

The overall purpose of this work is to support research being conducted within the 711 HPW/RHXS's Speech and Communication Research, Engineering, Analysis, and Modeling (SCREAM) Laboratory. Work in the SCREAM Laboratory includes the development of capabilities for speech recognition of foreign languages and speech-to-speech translation.

The focus of this task will be on MT, ASR, and TTS, especially Less Commonly Taught Languages (LCTL).

1.0 INTRODUCTION

This document provides a summary of work completed by SRA International for the HLT Group of 711 HPW/RHXS during the period 08 October 2008 to 31 March 2014 under ICER contract FA8650-09-D-6939, Task Order 0001.

Section 2.0 describes experiments and accomplishments in MT, ASR, TTS, Laboratory Corpora Support, as well as system administration efforts that support the research environment.

Section 3.0 provides summaries of conclusions drawn from the experiments and makes recommendations for future efforts.

2.0 EXPERIMENTS AND ACCOMPLISHMENTS

This section discusses experiments and accomplishments. As part of this, Section 2.1 covers MT, Section 2.2 covers ASR, Section 2.3 covers TTS, Section 2.4 covers Laboratory Corpora Support, and Section 2.5 covers Laboratory System Administration Support.

Under the ICER contract, work on MT, ASR, TTS, Spoken Language Translation Systems (SLTS), Named Entity (NE), and related technologies are included in multiple task orders that support the SCREAM Laboratory. In some cases, these task orders may focus on different languages of interest; however, many of the software utilities, algorithms, and procedures may be applicable to many languages and tasks. Some of the significant experiments and accomplishments described in this report may also appear in reports for other task orders as the work was split between multiple task orders.

2.1 MT

MT is one of the main research efforts of the 711HPW/RHXS SCREAM Lab. This section describes many research activities performed to support and improve the Laboratory's MT systems.

2.1.1. Champollion Extensions

The Champollion sentence aligner has been used in the lab to improve sentence alignment between parallel texts. As delivered, it provides dictionaries to aid in aligning Chinese-English and Arabic-English sentence pairs.

These dictionaries or seed translation lexica are necessary for acceptable performance when aligning sentences. Generating Seed Translation Lexicon (STL) for new language pairs can use data from multiple sources. These can be from existing dictionaries, LCTL Lexica, or even automatically generated word alignments from GIZA or related programs.

The Urdu STL was created from the LCTL Urdu Lexicon v1.0, with the addition of various morphologically inflected forms from the Humayoun morphological toolbox. This STL was used on data during the MT09 evaluation.

The Farsi STL was generated from in house data and is used to process the Dari-English portion of the Sada-e-Azadi corpus. There is some concern about differences between Farsi and Dari but there is enough commonality to allow for the use of a Farsi STL when sentence-aligning Dari.

For the Pashto-English STL, we started with the LCTL Pashto Lexicon v1.0, and then augmented with headwords extracted from the Raverty Pashto Dictionary. The combined STL is used in the processing of the Pashto-English portion of the Sada-e-Azadi corpus.

The Russian seed translation lexicon was generated from word alignments produced by GIZA when training on the International Workshop on Spoken Language Translation (WSLT) 2013 Russian-English data. Although not optimal, using this STL to realign the Russian-English portion of the Multi-United Nations (UN) corpus allowed a modest improvement when used as an additional source of training data.

2.1.2. MT09 Evaluation

Preparations were made for the MT09 evaluation, specifically for the Urdu-English systems. Tasks such as data sanitization, production of factored input and various data format tagging were performed.

2.1.3. IWSLT13 Evaluation

A number of tasks were performed in preparation for the IWSLT 2013 evaluation campaign to prepare data, write and integrate processing functionality into SMT2, design experiments, and run experiments for multiple language pairs.

English-French: The Massachusetts Institute of Technology Lincoln Laboratory (MIT-LL)/Air Force Research Laboratory (AFRL) system for the Technology, Entertainment, Designs (TED) Talks English-French system did not change significantly from the previous IWSLT system. Cross-Entropy filtering was added to augment the training data set used to train the translation and Language Models (LMs). For this language pair, the MIT-LL/AFRL system placed 7th, but with a final score still well above the provided baseline. Investigation is ongoing as to the manner of the score decrease relative to other submitted systems. Possible targets of interest include discrepancies in truecasing as well as further tuning of Moses parameters.

Chinese-English: Various Chinese word segmenters were evaluated in order to determine which yielded the best results. The Stanford Chinese segmenter yielded the best results for this data set (see Section 2.1.16).

Additional work was performed to investigate methods to decompose or simplify Chinese data using various tools but did not yield improvements.

For this portion of the MIT-LL/AFRL system, the above insights were used in further processing of the system. This system placed 3rd in the Chinese-English portion. Higher-scoring systems may have used hierarchical MT systems as opposed to the phrase-based approach that we pursued. Investigation on improving system performance is ongoing.

Arabic-English: The Arabic-English portion of the system was primarily run at MIT-LL. This system utilized combinations of MADA and AP5 to process the Arabic input data. This system placed 3rd in this portion of the evaluation.

Farsi-English: One of the new language pairs for this evaluation campaign is Farsi-English. Since the tokenizer typically used does not support Farsi stemming or tokenization, scripts were added to the SMT2 system to perform these tasks (Perstem, Pertok). However, these scripts did not offer a significant improvement over utilizing the baseline Moses tokenizer.

For this language pair, the MIT-LL/AFRL translation system placed first. This could be due to the novelty of the language pair for this year, such that other participants did not submit systems. However, we did leverage our knowledge of Arabic script languages to our advantage.

Russian-English: A good portion of the effort of investigating the data for IWSLT 2013 was spent on the Russian-English language pair. Lattice inputs with stemmed and other variants for Out of Vocabulary (OOV) words were created as an attempt to increase scores during optimization (see Section 2.1.15). This approach yielded a modest improvement over baseline, but was ultimately not used during system submission. A similar approach was tried that directly manipulated N-Best lists after decode in order to increase performance during rescoring, but this approach also did not yield significant improvements.

Research partners at MIT-LL revived old lexical approximation code (LexApprox) that allows substitution of unknown words for known words ‘near’ the spelling of the word in question. This approach yielded slightly larger improvements in score and was utilized during system

submission. This system placed 3rd among submitted systems, but the scores between systems were close enough to put our system within striking distance of the top systems.

2.1.4. Qahira

The Qahira word aligner was modified to use the A3Metric library to handle loading and writing word alignments, reducing duplication of code. Code was added to allow setting link probability when reviewed by a human operator. This code was supplied to Dr. Tamas Marinus at the Defense Language Institute (DLI) in an effort to evaluate word alignment quality for multiple languages.

There is interest from within the research community in making Qahira available for public use. Currently the best approach is to write a journal article, include the code as an appendix, and then get the amalgamated package cleared for public release.

2.1.5. Lader

In order to improve the target word order of various statistical machine translation systems, a software package was used to transform the input data of statistical MT systems to allow for more natural translated output. Lader is developed by Grant Neubig at Nara Institute of Science & Technology (NAIST), Nara, Japan. After various email exchanges, an English-French reordering system was trained and was able to produce an approximate one Bilingual Evaluation Understudy (BLEU) point improvement. Improvements in training time were achieved by parallelization of the reordering model training process at the expense of the sophistication of the learning algorithm. Reordering models were also trained for Russian-English and Chinese-English, but did not exhibit the same gains in BLEU score. Investigation into these results is ongoing.

2.1.6. LM Interpolation

In order to provide larger, more complete LMs for use in statistical MT systems, a method has been devised to statically interpolate smaller component LMs into one larger LM. Existing LM toolkits (Stanford Research Institute LM Toolkit (SRILM) and Massachusetts Institute of Technology LM Toolkit (MITLM)) can perform this work, but typically load some or all of the component models into memory simultaneously. Our approach allows for only having one component model loaded at once, saving memory and allowing for generation of larger interpolated models. Although currently unpublished, results from this work will be likely be submitted to Association for Computational Linguistics (ACL) 2014,¹ Workshop on Machine Translation (WMT)14,² or Empirical Methods in Natural Language Processing (EMNLP)14³ [1].

2.1.7. SMT2 Improvements

Improvements to the SMT2 Statistical MT system were made during the duration of this work in the areas of word alignment, non-standard decoding support and cross-entropy filtering. Additional integration of various processing scripts was also performed.

Word Aligners: In addition to the previously integrated Posterior Constrained Alignment Toolkit (PostCAT) and Nile word aligners, Phrasal Inversion Transduction Grammar (ITG)

¹ <http://www.cs.jhu.edu/ACL2014>

² <http://www.statmt.org/wmt14>

³ <http://emnlp2014.org>

Word Aligner (PIAlign) and FastAlign were integrated into SMT2. The Bilingual Alignment (BIA), and RegAlign word aligners as well as a version of GIZA with a complete implementation of a non-deficient International Business Machine (IBM) Model 4 were also investigated.

Word Alignment Filtering: Two approaches to filtering word alignments were developed over the course of this task. The earlier one – A3Threshold, focuses on filtering sets of alignments using the reported score. Experiments run with this kind of filtering used the GIZA score – other aligners typically do not utilize the score field. Filtering in this manner does reduce available alignment, putting more work on the phrase extractor and can remove useful information from the phrase table. The hope is that any useful data removed is offset by the larger amount of removal of poor-quality alignments. In practice, this approach can produce gains in BLEU score, but is limited to smaller systems with less than 100,000 lines of parallel text.

The other approach is to consider the alignments from a series of aligners (typically 6-8) – if a given word pair occurs in the set of word aligner outputs above a threshold of n , that link is preserved. These voted-upon links are then output and used for phrase extraction. While this approach does yield small increases in BLEU score, the benefit lies in the reduction of phrase table size by upwards of 60% without degradation of system performance. Although currently unpublished, results from this work will be likely be submitted to ACL 2014,¹ WMT14,² or EMNLP14³ [2].

2.1.8. Language Resource Identification

Language resources needed to support MT include parallel text for training, morphological tools for stemming and part of speech tagging, and general information about the scripts and grammar of the languages under study.

Identification of Parallel Text Resources: Online sources were found for additional parallel text and word lists for Pashto. The Institute for War & Peace Reporting⁴ was identified as a source of parallel text, with about 100 news articles in English/Dari/Pashto and ongoing postings. A dictionary of Pashto military terms⁵ was also identified. In addition, various flashcard websites were noted that could be used to extend the Pashto lexicon by hundreds of entries.

Online parallel text resources for Urdu were reviewed. Some previously identified sites were not as useful as expected; in particular, Central Command (CENTCOM) site creates pages for Urdu translations of its articles, but a large number of these articles are not actually translated.

References to various parallel corpuses were noted for Farsi such as the Hamshari newspaper corpus, but these have limited availability.

The Russian National Corpus website was reviewed, and it was determined that only a subset of this data is freely available.

The *Journal of Onomastics* was identified as a possible general source for named entity lists.

Identification of MT Resources for Russian: The highly inflected morphology and relatively free word order of Russian affects the application of MT tools. A review was made of available

⁴ <http://iwpr.net/>

⁵ <http://www.torresco.com>

Russian resources for MT, including parallel text, lexica, and tools for MT such as part of speech taggers and stemmers. The Leeds website on Russian statistical taggers and parsers⁶ was identified as a good index of existing MT tools. A list of other Russian resources is included in Appendix A.

Language Study: Work on this project entailed review of the script, grammar, and basic vocabulary of a variety of languages, including Pashto, Dari, Tajik, Hindi, Urdu, Chinese, and Russian.

2.1.9. Character-Based Processing

Character-based processing can improve MT by reducing variation in spelling and tokenization.

Spelling Normalization: Spelling normalization addresses variation in the data due to dialectal differences, spelling errors, and problems in the encoding of text.

Spelling Correction and Dialect Normalization with Variant Conversion (VARCON): Spelling and typographical errors cause word-level variation that adversely affects MT by increasing the data sparsity.

British vs. American English: The Urdu/English parallel text contains a particular dialect of English that includes British English spellings and expressions along with American English spellings. The VARCON dialect conversion program [3] was adapted to record and correct common spelling errors in the Urdu/English data. The VARCON program uses lists of word pairs, allowing us to automatically change movement to movement (in spelling correction), or colour to color (in dialect conversion). British English spellings were converted to the slightly more common American English spellings.

Urdu Splitting and Compounding: Another type of spelling error involves splitting and compounding; this is more difficult to correct, as it relates words to phrases. Splitting occurs when writers place spaces within a word; its counterpart, compounding, occurs when writers combine two or more words with no intervening spaces. Such spacing variation may be derived from errors, from cross-language differences, or it may reflect legitimate variation within a language (English examples would be snowfall vs. snow fall, deVille vs. de Ville). In Urdu, compounding is common when the final letter of one word is a non-joiner, so there is the appearance of a break in the script. Cross-language spacing variation is tractable, if it is consistent and if there are enough examples. While it may be an error for Urdu writers to omit the space in جوبائیڈن /jwbA'ydn/ “JoeBiden,” if this occurs frequently, then the MT system can learn the mapping. Spacing variations within infrequent words, however, create problems for the MT. Urdu splitting and compounding was addressed in this effort as a morphological process; see Section 0

Language Identification with Aspell: The spelling correction program Aspell can be used to detect sections of wrong-language text within a file. See Section 2.1.10 for details on the use of Aspell to detect problems in the English file of the Common Crawl dataset.

Encoding Problems: Character encoding issues were detected in a variety of datasets. An existing program, “KFindExtrasOrderedEnglish.java,” was applied to find control characters

⁶ <http://corpus.leeds.ac.uk/mocky/>

and also list characters from outside of the English character range for further examination. See also specific encoding solutions described in Sections 2.1.9 and 2.1.10.

French: Out-of-range characters were detected in French lexicon files. It was determined that these characters mostly come from named entities from other languages (e.g., Schrödinger with the umlauted vowel), and did not need to be corrected.

Urdu: In the MT08 Urdu data, some punctuation marks were found to be encoded with odd characters. For example, the sequence THE COMMUNIQUE STATED was encoded as THE COMMUNIQUE³TATED. A program was written, “*mt08convert.pl*” that maps such sequences on a case-by-case basis. The “*mt08convert.pl*” program also removes instances of the control character FEFF, which may be the residue of a byte code marker from the start of individual files.

Russian: The Russian/English Common Crawl text contains a variety of encoding problems that reduce its usefulness as parallel text. There are some French and German words in which accented characters are represented with Cyrillic characters, for example, й=é в=â и=è κ=ê φ=ô. This is due to a conflict between UTF8 and International Organization for Standardization (ISO) encodings. (This is the reverse of the ISO vs UTF8 encoding problem in the Russian text discussed below in Section 2.1.10).

Example French sentence with encoded accents:

M. et Mme Bouvier et l'йquipe de professionnels seront heureux de vous recevoir dans ce beau chateau du 18йme entre mer et forkt au coeur de la cφte d'Emeraude.

corrected to:

M. et Mme Bouvier et l'équipe de professionnels seront heureux de vous recevoir dans ce beau château du 18ème entre mer et forêt au coeur de la côte d'Emeraude.

Two programs were written to correct these mappings, “*normalizeAccentsEncodedWithCyrillic.pl*” (for the English text) and “*normalizeAccentsEncodedWithCyrillicInBorrowedWords.pl*” (for borrowed words in the Russian text). The second program only applies to words with Latin characters, so as to prevent the alteration of actual Russian words.

There are false positives with this approach when Russian and English words have been run together.

Examples of false positives in converting Cyrillic within Latin words:

png|thumb|500px|Последние >> png|thumb|500px|Îîñěăăîéă
Professionalзапасные >> Professionalçàîăñîûă

There are also several additional strange encodings in the Russian text, including some that use two characters to encode a single character, such as Ðµ for Cyrillic e. A conversion program could be written for each mapping; however, human supervision would be needed to determine which mapping to apply.

Tokenization: Tokenization is used to separate words from punctuation, allowing the MT system to recognize punctuated and unpunctuated words as instances of the same token.

General: Some general problems were addressed in the application of the tokenization program, including the treatment of special characters and the treatment of sequences of punctuated elements.

Escaping HTML: The new Moses system replaces (escapes) apostrophes and certain other punctuation marks with HyperText Markup Language (HTML) before translation and restores (de-escapes) them after translation (e.g. the punctuation mark ' becomes '). Since some SCREAM Lab processing takes place at intermediate steps, various tokenization scripts such as “*removeHTML.perl*” had to be revised to anticipate these new forms.

Global Rule Application Problem in Moses Tokenizer: A problem was detected with global rule application in the Moses tokenizer. This affects a small number of sentences in the Chinese, French, and Russian data, in which commas are used to separate individual letters, as in this example:

想想看 基因编码有四个字符：A,C,G和T。

Think of this: we have a four-letter genetic code: A, C, G and T.

The tokenizer uses a global Perl matching rule to identify sequences of letter-comma-letter and put spaces before and after the comma. However, after each match, the rule continues by examining the character after the last matched element. This means that the comma rule will only see every other comma in a sequence like A,B,C,D,E:

applies at * A,B,C,D,E,F
 * * *
 A,B,C,D,E,F >> A , B,C , D,E , F

A revised version of the tokenizer was created that prevents this problem by splitting the rule into two steps.

A similar problem was found for hyphen-marking, in which the tokenizer optionally encloses hyphens in @ characters. If there is a sequence of single characters separated by hyphens, hyphen-marking will only apply to every other hyphen. Here, a two-step solution cannot be applied. Instead, the perl variable pos(string) should be used to reset the character search.

French: The French data present tokenization problems involving the treatment of contractions, the interaction of casing with word order, and French punctuation conventions.

Contraction Expansion: Recent French/English experiments were reviewed for tokenization problems. Contraction expansion was identified as a potential problem. French usage requires the elision of the vowel in the words *je me te se ce que de le* and *la* when they are followed by a vowel-initial word. The SMT2 tokenizer expands these French contractions in both training and test data in order to reduce data sparsity; however, this process is not reversed for the final output of translation. A re-contraction program, “*Recontract.java*,” was written that forms contractions of the French pronouns when they occur before a vowel; the detokenizer can subsequently adjust spacing appropriately. An example is shown for the for *j'ai*, a contraction for *je ai* 'I have.'0

<u>Input</u>	<u>SMT2 Tokenizer</u>	<u>Recontract.java</u>	<u>Detokenizer</u>	<u>Final Form</u>
j'ia	je ai		No change	je ai
j'ai	je ai	j'ai	j'ai	j'ai

A truecasing program then capitalized certain words in the output according to learned patterns, such as the capitalization of the first word in a sentence. Unfortunately, this recontraction and truecasing process failed to improve the overall MT scores.

The “*Recontract.java*” program was revised to correct some errors in case-sensitive matching. Further research on French usage identified some exceptions to the elision process, for possible inclusion in future versions of “*Recontract.java*.” For example, while *la + unique* becomes *l'unique*, *la* remains uncontracted in the phrase *à la une* “on the front page.” These problems were considered minor.

In general, recontraction fails to improve scores because of the potential for mis-translating the word that triggers the contraction. Contraction occurs if the main word begins with a vowel. If the target translation begins with a consonant, then it requires an uncontracted form, and recontraction with the translated, vowel-initial word reduces the match between hypothesis and target.

For example, the word *eco-system* was correctly translated as *écosystème*, and recontraction improves the match with the target. In contrast, the word *editor* was translated as *éditeur* instead of the target *rédacteur*; and, since the hypothesis begins with a vowel, it triggers recontraction, reducing the match with the target, a consonant-initial word that should not trigger contraction.

	<u>Tokenized</u>	<u>Recontracted</u>	<u>Detokenized</u>	<u>Notes</u>
Output	le écosystème	l' écosystème	l'écosystème	gained match
Reference	l'écosystème			
Output	le éditeur	l' éditeur	l'éditeur	lost match with <i>le</i>
Reference	le rédacteur			

Truecasing: The results of truecasing were examined. A program, “*checkFirstWords.pl*,” was written to look for changes in case of the first letter in the line. There were nine lines in which the initial word was lowercase, which causes a mismatch after truecasing. Another program, “*tcDelta.perl*,” was written to look for case changes in any matching words when comparing the truecased output with the reference. This did not identify any particular problems with the truecasing process.

	<u>Lines</u>	<u>Words with Wrong Case</u>	<u>Average Wrong Words per Line</u>
2011 Truecased	818	313	0.278
2012 Truecased	1124	205	0.251

Truecasing improves the match with the reference if the output begins with the same word as the reference. Otherwise, truecasing can remove an existing match. For example, the phrase *you see* was translated as *vous voyez*, but the reference has *Et vous voyez* (literally, “and you see”). Truecasing the initial word *vous* reduces the match to the reference

	<u>Original</u>	<u>Truecased</u>	<u>Notes</u>
Output	vous voyez	Vous voyez	Lost match with <i>vous</i>
Reference	Et vous voyez		

One systematic source of mis-cased words is the French requirement that nouns be introduced with a determiner. For example, there is an English sentence that begins with *Secrets*; the MT into French begins with the (French) word *secrets* in turn; but the French reference exhibits the required determiner in *Les secrets* ‘the secrets.’ This means that capitalizing the S in *Secrets* actually takes the output further away from the French reference.

	<u>Original</u>	<u>Truecased</u>	<u>Notes</u>
Output	secrets	Secrets	lost match with <i>secrets</i>
Reference	Les secrets		

Another source of case changes is the use of borrowed English in quotations or book titles. The French reference may retain the original English, with capitalization, as in *The Great Gatsby*, while the MT output translates some of these words, in lowercase: *le grand gatsby*.

Names, when unknown in the training data, can appear mis-cased, e.g., *John Keats* was translated as *John keats*. Website names lose their unusual capitalizations in translation: *DonorsChoose.org* becomes simply *donorschoose.org*. Other names which are ambiguous with common nouns can end up translated in error: *Hazel* becomes French *noisette* “hazelnut;” the acronym *NICE* (*National Institute for Clinical Excellence*) becomes *bien* “good.” Case mismatches also occur when the reference sometimes fails to capitalize appropriately (e.g., the reference lists the name *Henry ford* with a lowercase *f*).

Other Tokenization Issues: Certain French punctuation patterns can affect the performance of tokenization and detokenization. French punctuation traditionally uses angled quotation marks called guillemets: « » instead of the straight quotation mark " used in the English data. Traditional French usage also requires a space before the punctuation marks, ?!;#\$ and %, as well as a space between the guillemets and the quoted material. These spaces are removed by the detokenization program.

A program was written, “*addSpaces2FrenchPunct.perl*,” to insert the required spaces for these punctuation marks. However, the data display both traditional French usage and English-style spacing, so it is unclear whether normalizing to the French usage will be useful. The 2012 data

tend to follow the traditional guidelines, while the 2011 data show more Anglicized usage. The training data has mixed usage.

	«	"	<u>Space Before ?</u>	<u>No Space Before ?</u>
2011	10	67	11	31
2012	56	5	57	3

There is also a tokenization problem involving the use of musical notation ♪ and ♫ in the TED Talk data when a song is performed. For example:

♪ And that is all ♪ ♪ that love's about ♪ ♪ And we'll recall ♪ ♪ when time runs out ♪ ♪

These sections may contain multiple sentences punctuated by notes instead of periods. A program was written, “*fix-notes.pl*” that replaces notes with periods when a capital letter follows, and removes notes that occur without following capitalization.

Chinese: The tokenization of Chinese requires attention to special characters, the presence of English within the original Chinese text, and the segmentation of untranslated Chinese words in English text.

Conversion between Full-Width and Standard Characters: The Hong Kong newswire dataset was found to have unusual characters. A program was written, “*ChineseNormalize.pl*,” to detect non-Chinese characters and to convert full-width forms to more common forms.

Additions to Normalizer: The Chinese TED Talk data contain files that process properly when word segmented, but cause errors in the MT process when character segmented. Examination showed that the only difference between the well-behaved word-segmented files and the problematic character-segmented files was the treatment of hyphenation, in which the dash characters are replaced by multiple hyphens. Normalization of the characters 2013 – (en-dash) and 2014 — (em-dash) to single hyphens prevented this problem.

Addition of English Tokenization to Character Segmenter for Borrowed Words: The Chinese character segmentation program, “*cseg2013.pl*,” was extended to add tokenization of English sequences. Code from the Moses tokenizer was applied in order to properly space punctuation and contractions when English text is detected.

Prevention of Recombination of Chinese Words within English Text: The latest version of the Moses detokenizer combines sequences of Chinese words. This is the desired result for Chinese output, which is typically not word-segmented, but not for English output that happens to contain Chinese OOV words. In the latter case, word segmentation should be preserved to enable analysis of the individual OOV words. A language flag was added to the Chinese concatenation clause of the detokenizer, so that concatenation will only apply when the specified language is Chinese, Japanese, or Korean. For example, the English sentence below contains two Chinese OOV words, 情懷 and 痴心妄想. The revised detokenizer keeps these as separate words.

Original

and i 'm sure there 's definitely some of 情懷 痴心妄想 or nostalgia .

Old Detokenizer Output

and i'm sure there's definitely some of 情懷痴心妄想 or nostalgia.

New Detokenizer Output

and i'm sure there's definitely some of 情愫 痴心妄想 or nostalgia.

Russian: Problems were found in the Russian data that affected tokenization. These included errors in the list of elements that can occur with period punctuation (non-breaking prefixes) and errors in which Russian words were written in mixed alphabets.

Non-Breaking Prefix Additions: The Russian nonbreaking prefix list used by the Moses tokenizer was found to be incomplete. This list serves as a guide to the tokenizer by enumerating words that may be followed by a period without indicating the end of a sentence. The Russian initials А-Я were added and other abbreviations found in the data were identified for possible inclusion in future lists (for example, им. which is the abbreviation for имени meaning “by the name of”). It was also noted that some English initials (J, H, W, Q, X, Y) do not have obvious translations via Cyrillic initials. For example, J.K. Rowling becomes дж. к. роулинг, using the two characters дж /d ž/ to phonetically create the sound of J, so дж should be added to the nonbreaking prefix list as well.

Russian translations for sentences containing these English initials were examined. The initial J was translated as дж, as mentioned above; W was translated as either В /v/ or У /u/. Translators also tend to avoid the problem, by spelling out the full word, omitting the initial, or by keeping the English characters, as illustrated below. Note that some examples also exhibit case endings in –a and –u.

<u>English</u>	<u>Cyrillic</u>	<u>Transliteration</u>	<u>Notes</u>
J.K. Rowling	дж. к. Роулинг	dž. k. rouling	two characters for J
W.H. Auden	В.Г.Ауден	v.g.auden	w > v, h > g
W.B. Yeats	У.Б.Йейтса	u.b.yeytsa	w > u
George W. Bush	Джордж Буш	džordž buš	w. omitted
J.D. Hooker	Джозефу Хукеру	džozefu xukeru	h > x, full name used for J
I.Q.	I.Q.	---	Latin characters retained

Non-Breaking Prefix Correction: Newlines vs. Carriage Returns: The nonbreaking prefix list for Russian contains Windows-style carriage returns 000D instead of Linux-style line feeds 000A, as expected by the Moses tokenizer. The presence of the carriage return characters prevented the recognition of the prefixes during tokenization. Two scripts were written, “*countCarriageReturns.perl*” and “*removeCarriageReturns.perl*,” to identify and fix this problem in the Russian file and two other prefix files (.ca and .sl).

Mixed Latin and Cyrillic Characters: An examination of the Russian training data detected a few hundred words that were incorrectly entered with a mix of Cyrillic and Latin characters, such as она “she” which was sometimes typed with a Latin o and Latin a, instead of the Cyrillic o and a. While visually the same as correctly spelled words, these words are treated as separate tokens in the MT process. A program was written, “*reportMixedLatinAndCyrillicCharacters.pl*,” to identify these problems, and another program, “*cleanMixedLatinAndCyrillicCharacters.pl*,” was written to restore these words to the correct script. Exception handling was included to prevent the program from changing hyphenated words consisting of one word in each script, such as МР3-плеер “MP3-player.” The program was extended to normalize words that contain stress marks that are not normally indicated in Russian, e.g., большо́ую ‘big.’

These mixed character words rarely occur in the current dev and test data, and applying “*cleanMixedLatinAndCyrillicCharacters.pl*” did not improve the translation results (see Section 2.1.15).

An examination of the augmented phrase tables *maxl5.pt* and *maxl5+lexapprox.pt* revealed the presence of sequences such as: *nbsp* and *ampnbsp*; which are probably residue from HTML encodings for non-breaking spaces. There were 1253 words with attached HTML residue in the phrase table, which contributes to data sparsity. In addition, since the HTML forms are Latin character sequences, when the base Russian word is shorter than the HTML sequence, the “*cleanMixedLatinAndCyrillicCharacters.pl*” program attempts to normalize the word to all Latin characters.

A revised program was created, “*normalizeMixedLatinAndCyrillicCharactersEscaped.pl*,” adding a step to screen out the sequence *nbsp* and also the Moses tokenizer escape sequences (' " & | < > [[). Words containing these elements are not considered for character conversion. Further analysis revealed about 300 words with the sequence *ampquot*, which should also be screened out.

2.1.10. Alignment

In order for the language data to be useful for pattern matching, it must be aligned by sentences and then by words.

Sentence Alignment: The training data for MT is supposed to be presented as parallel text, in which each sentence in the foreign language matches a corresponding sentence in the English data. When the sentences are misaligned, the MT system learns false translation patterns.

Detect Misalignment by Word Counts and By Comparing Numerals: There was an alignment problem in the Urdu reference translations, in which documents had been reordered according to the document id number. In order to identify instances of mis-aligned sentences, a program was written, “*KFlagLines.java*” that recorded lines in which the Urdu and English translations differ in length by an amount greater than a specified number.

A review was made of the performance of the Vanilla aligner on the Joint Research Centre (JRC) Acquis database. A program was written, “*KNumberCheckAlign.java*” to identify potential mis-alignments by finding unmatched numerals across a language pair.

Detect Wrong Language via Character Ranges or Spelling Correction Dictionaries: Some datasets contain sections of wrong language text. This can be detected by checking for characters outside the Unicode range for the current language, using the existing program “*KFindExtrasOrdered.java*,” or by using a spelling correction program with a dictionary for the specified language. The Common Crawl dataset for Russian and English parallel text contains significant amounts of wrong-language text in both the Russian and the English files.

A program was written, “*findtwinsnonum.pl*,” to check for matching sentences in the English and Russian IWSLT training data. Matching sentences sometimes indicate sections that were mistakenly left untranslated, leaving English text in the Russian file; this can lead to English-English entries in the phrase table, and prevent translation of those words into Russian. The program excluded digits, since these can legitimately match across languages. Instances of matching or partial matching were primarily due to quotations, names, or websites (which should

remain in the original English), as well as foreign language sections that are neither English nor Russian.

Russian	Поют : We can do it . We can do it .
English	Chanting : We can do it ! We can do it !
Russian	Huffington Post , the Washington Post , the New York Times – все так или иначе балуются персонализацией .
English	Huffington Post , the Washington Post , the New York Times -- all flirting with personalization in various ways .
Russian	Ciao bellos!
English	Ciao bellos!

In addition to English quotations in the Russian file, the Common Crawl contains sections with the wrong languages (such as Ukrainian instead of Russian) and also sections with bad encodings. A series of programs was created to correct these problems. Altogether, this process removed 99,621 bad lines from the total of 878,386 lines of parallel text; 94,062 lines were identified as non-Russian lines in the Russian text, and an additional 5,559 lines were identified as non-English lines in the English text.

For the Russian text, the program “*cleanRussianCommonCrawl.pl*” first removes non-Russian sections. The program checks for words with Latin characters, and removes lines that have fewer than half of the words in Cyrillic characters. The program also removes lines that contain the Ukrainian characters і ї р є, which are not found in Russian. Exception-handling was added for the Ukrainian character І used in Roman numerals, and for the Ukrainian character р used in Latin words to encode apostrophes.

Examples of Ukrainian sentences to be removed from the Russian text

Важко сказати що саме приваблює слухачів: чи то енергійна українська музика, чи то красиві тексти чи то просто харизматичний Олег Скрипка.

Цього журналіста кинуть за ґрати чи вишлють з України?

Example of Roman Numeral with Ukrainian І within a Russian sentence

Мы рады предложить Вам на выбор любой из своих замечательных залов – большой зал ресторана, малый зал или уникальный «Летний сад» со старыми платанами, цветами, статуями XIX века и бьющим фонтаном.

Example of Ukrainian р used to encode apostrophe in non-Russian words

Песня The Kelly Family Irm So Happy представлена вам Lyrics-Keeper.

Finally, the program “*cleanRussianCommonCrawl.pl*” detects and recovers Russian lines that have been encoded in accented characters (via confusion between ISO and UTF8 encodings). The actual characters are restored by shifting the codepoint by adding 848.

Example of recovered ISO mapping of Cyrillic characters

Њїđàâêà ĩĩ âĩđĩâàì Đĩññèè è ièđà. >> Справка по городам России и мира.

The existing program, “*normalizeMixedLatinAndCyrillicCharactersEscaped.pl*,” was next applied to normalize the spelling of words with mixed Latin and Cyrillic characters. The normalization converts the word to all-Cyrillic if possible, or to all-Latin as a second choice.

HTML encodings for ' " nbsp etc. are first set aside, so they will not be included in the Latin character count, and then restored after the spelling normalization process.

For the English text, the program, “*removeForeignLinesWithParallelText.pl*,” was written to remove lines with primarily non-Latin words. The program first checks for words with non-Latin characters and removes any lines that have fewer than half of the words in Latin characters. This clears out Russian, Greek, etc. Then, the program checks for words containing accented characters like â; if more than half of the words contain such characters, the line is removed. The parameter must be set to strike a balance between allowing accented characters in borrowed words and names, and identifying sections in languages like Polish and Latvian that contain some accented characters.

Example of a sentence removed when it should be kept: too many accents in borrowed words:

168780\\The seven principal islands of Lofoten are Austvågøy, Gimsøy, Vestvågøy, Flakstadøy, Moskenesøy, Værøy and Røst. << REMOVED

Example of accents used to correctly ID and remove a foreign language sentence:

321952\\Šeit varat atpūsties, malkojot kādu dzērienu un papļāpāt ar kolēģiem par dažādiem ar dzīvi saistītiem jautājumiem. << REMOVED

Example of a sentence that is kept when it should be removed: not enough accents to ID this as a foreign language:

321950\\Esiet aicināti apspriest riepās, diskus un citu produkciju. << KEPT

Because of these difficulties, this step was replaced with foreign-language detection via the spelling correction program, Aspell. The program, “*aspell-check-encodings-english-4min-parallel.pl*,” uses Aspell to check each word in a line against an English dictionary; more than half of the words must be recognized for the sentence to be maintained as an English sentence. Compared to the accent-based approach, this method removes an additional 6,235 non-English lines, for a total removal of 11,794 non-English lines. An additional 94,062 non-Russian lines are removed, for a total of 105,856 lines removed, which is 12% of the original 878,386 lines.

The spelling correction approach can have both false positives and false negatives. Some of the newly removed non-English lines are false positives, involving a high proportion of borrowed words, named entities, or brand names that are unknown to the Aspell spelling correction program. This method appears to work best with a threshold requiring 50% or more recognized English words, thus allowing some unrecognized words within an otherwise English sentence. On the other hand, some foreign text contains short words that coincide with valid English words (German *die* “the,” Dutch *van* “of,” French *on* “one/he”); these are accepted as English by the spelling correction program, and can cause the program to accept a sentence that should be rejected. The program was therefore modified to discard from consideration any words shorter than four characters.

Word Alignment: During the training of a MT system, words from parallel sentences are automatically matched, or aligned, and these word alignments are used to derive the phrase table entries for translation of new documents. Some automatic processes can improve word alignment quality. Alternatively, having a person hand-align or hand-edit machine-aligned sentences creates better word alignments and therefore better phrase table entries.

Automatic Word Alignment: During this effort, investigations of improvements to automatic word alignment included prevention of links between words and punctuation, and changes in distortion limit.

Automatic Removal of Links between Word and Punctuation: To improve word-level alignment, a utility program was added to the A3Metric suite of word-alignment tools that removes links between word tokens and punctuation tokens. This should improve the quality of the phrase table that is derived from the word alignment.

Distortion Limit for Russian: An alignment metric based on distortion was added to the A3Metric that calculates distortion by the relative word position of a linked pair of words in their respective sentences. This metric is calculated on both a per-sentence and per-corpus level. The intent of this metric is to arrive at a starting value for a near-optimal setting for the word distortion limit parameter in Moses.

Some distortion of word placement is expected for the English/Russian language pair, because Russian omits words found in English, such as determiners and some subjects. For example, the word *raises* is third in the English sentence below, but its counterpart *возникает* “raises” is first in the Russian sentence, due to the omission of the conjunction and the subject.

and it raises a question : what changed ?

возникает вопрос : что изменилось ?

literally: raises question : what happened ?

Hand Alignment: During this effort experiments were performed using both existing hand-aligned data (gold standard alignments) and new data that is hand-aligned by language specialists.

Sources of Existing Gold Standard Alignments: An additional source of French/English hand-edited word alignments was identified; the Graça et al. collection of 100 hand-aligned Europarl sentences⁷.

Plans for a Combined Hand-Aligned Database for Russian, French, and Chinese Sentences: Plans were made to coordinate the hand editing of word alignments for French, Chinese, and Russian, in order to create a database with hand alignments over the same sentences in each of these languages. The TED Talk files were examined to identify overlapping talks in the French, Chinese, and Russian datasets. An initial review of the data indicates a need to resolve differences in tokenization and sentence segmentations.

Cost of Hand Alignment: A review was made of the hand alignment process, including the cost and time required to develop hand-aligned files in additional languages. Hand-editing of existing alignment files is paid according to the number of links changed, as recorded by the program “*KCountA3Changes.java*.” This program was revised to exclude payment for changes to links from the NULL node, since such changes do not give us any additional information (if the editor does not remove these links, they are removed automatically at the conclusion of hand-editing).

Metrics to Prioritize Sentences for Hand Alignment: The hand-aligned Urdu files were analyzed in an attempt to discover characteristics of an automatically-aligned sentence that make it most worth sending to a human editor for correction.

⁷ <http://www.l2f.inesc-id.pt/resources/translation/>

The following metrics were examined: percent of links changed, sentence length, the automatically-assigned alignment score, and the percent of links matching lexical entries. The hope was that one of these metrics could be used as a diagnostic to identify poorly aligned sentences. No clear pattern emerged from this analysis.

This process did help identify some problem sentences, such as sentences truncated at a decimal point, indicating poor sentence alignment, and some sentences that had been skipped by the human editor, as evidenced by a lack of edits despite the fact that none of the original links were validated by the lexicon.

Guidelines for the Hand Alignment Process: Published alignment guidelines for the hand-editing of word alignments were reviewed, with the goal of developing guidelines for SCREAM Lab annotators. In general, the guidelines are tailored to the syntax and morphology of the languages under study. The published guidelines also differ in their definitions of S (sure) and P (possible) links, and in the directions given for the alignment of auxiliary words and phrasal idioms. Some authors apply S links to those word pairs that will always be good translations of each other, regardless of context, and use P links for word pairs that are matched only in the current context; other authors use P links to specify ambiguous word alignments within the current sentence. If a sentence has a determiner or an auxiliary verb not present in the translated sentence, some authors prefer to leave that element unlinked, others link it to the main word with a P link, and some use an S link. For idioms, in which two phrases match but the component words do not match, some authors link all words to all words with P links, while other authors use S links, and some define phrasal links, with potential individual word-pair links within the phrase.

2.1.11. Improvements to the Urdu Lexicon

The LCTL Urdu language pack [4] provides an Urdu-English lexicon. This lexicon was developed from several online dictionaries, so it contains errors, omissions, and inconsistencies. Previous work had addressed error correction; the current effort adds lexical entries for missing words. An examination of the MT data identified words that were not present in the lexicon; attempts were then made to find definitions of these, either from the context of the translation, or in consultation with a native speaker.

Missing Monomorphemic Words: Some of the missing words are found in the lexicon with different morphological endings; these are addressed in Section 2.1.11. This section addresses missing words that are not morphologically complex.

Manual Definition: Some unknown words were identified by context, and confirmed by comparison across sentences. Borrowed words and names appeared in transliteration, and could be identified by their sound patterns or by the presence of a corresponding named entity in the parallel English text. Other unknown words could only be identified by a native speaker consultant.

Automatic Definition: Two techniques were explored for automatically determining the definition of unknown words in the context of the reference translations, when these are provided. One method, intersection translation, compares the English reference sentences for all the sentences containing the unknown word, and finds the English words they have in common. For example, there two sentences that are output with the untranslated word لابسٹر /lAbstr/.
MT Output: The age of لابسٹر can be derived from it .
Reference: The age of a lobster can be estimated from its weight .

MT Output: The immediate and لاسٹر publicity for the pictures to use their started .
Reference: and started using the lobster 's pictures immediately for publicity.

The set of words found in both references is {*the, lobster*}, which includes the correct translation. A human editor can correctly identify the untranslated word as *lobster*; the program will need some additional constraints to be able to choose the correct word.

The second method takes an unknown word from a single sentence, and uses the English reference translation words as a filter to restrict possible candidates for the transliteration of the OOV word. This improves the ability of the transliterator to find the correct English translation when there are multiple words that match the sound pattern (See Section 2.1.14). For example, one sentence includes the name, *Professor Elizabeth Gardner*. The MT correctly translates *Professor* and *Elizabeth*, but leaves *Gardner* in the original Urdu spelling, گارڈنر /gArɖnr/. The transliterator, given /gArɖnr/, identifies several possible English words based on the pronunciation dictionary. Without further information, the transliterator prefers the most frequently occurring choice, *gardener*. By comparing the choices with the English reference, the program can instead select *gardner*.

Missing Inflected Forms: Frequently, the missing words were inflected forms of common words, as the LCTL lexicon tends to provide only the basic form of a word. The lexicon is missing the inflected forms of common verbs like "go," "do," "have," as well as plurals and oblique forms of common nouns.

For example, the lexicon has one entry for "fisherman" in the nominative singular, مچھیرا /mčhyrA/; but the parallel text also includes forms for "fisherman," oblique singular, مچھیرے /mčhyre/ and "fishermen," oblique plural, مچھیروں /mčhrwn/.

An algorithm was designed to introduce lexical information from the Humayoun Urdu morphological analyzer [5] into the LCTL-provided Urdu lexicon. First, the Humayoun data was converted into Urdu script, using the Humayoun transliteration tables. Then, the Humayoun entries were entered into the LCTL lexicon, using the LCTL definition for the base form when available.

In addition to the inflected forms of Urdu words, there were some unknown words that turned out to be combinations of English words and Urdu morphology. For example, *propagand-on* "propaganda + plural." These were defined individually.

Other Problems for the Urdu Lexicon: The nature of the Urdu language creates other problems for the lexicon. These include the treatment of phrases and compounds and variations in spelling. Additionally, the LCTL lexicon has inconsistencies in its annotation of morphological information.

Borrowed Phrases: Some unknown words reflect the borrowing of phrases from Farsi and Arabic, in which the individual words may lack independent meaning for Urdu speakers (much as an English speaker may know the French phrase, *laissez faire*, but not know the meaning of the individual words).

Splitting and Compounding: Some borrowed English words are written as multiple words in Urdu, leading to the creation of "words" that are difficult to define. For example, Urdu borrows the English word *associated* but spells it as two words, ایسوسی ایٹڈ "associ" "ated," creating an apparent word, ایٹڈ "ated."

Urdu also contains native forms like پسندوں /psndwn/ “likers” which has the appearance of a separate word but is primarily used in combination with other words to form many political words, such as “equilibrium likers” meaning moderates, “freedom likers” meaning separatists. See Section 0 on the normalization of split and compound words in Urdu.

Spelling Normalization: Spelling errors were identified and corrected in the LCTL Urdu lexicon. An interface was written, “*aspell-correct-lexicon2.pl*,” to use the Aspell spell-checking program to identify misspellings in the English translations. Another program was written to detect instances of multiple diacritics in Urdu words. These errors are not visually detectable, since the diacritics are written in the same space as the preceding letter, but they will still cause the word to be considered as a distinct form for computation.

Meta-Data: Inconsistencies were noted in the meta-data recorded in the LCTL Urdu lexicon. Morphological tags were sometimes contradictory within a single entry, and multiple versions of the same tag were used with different entries. For example, the list of inflectional features for a single entry includes the redundant forms, *Adjective+mascadj+masc*. Singular nouns were annotated by

Sg, sg, or sgnoun, while plural nouns were annotated by *Pl, Plu, or pl*.

2.1.12. Morphological Processing

Cross-language variation in morphology introduces complications for MT. This can be addressed with morphological processes such as part-of-speech tagging, named entity tagging, stemming, splitting of compound words, and the conversion of digits to words.

Part-of-Speech (POS) Tagging: POS tagging identifies classes of words; these can then be used in further processing, such as factored translation, the generation of inflectional endings, or the selection of syntactically matching phrases when creating paraphrases.

English: Several English POS taggers and morphological analyzers were evaluated, including Morpha, XTAG, and Maximum Entropy POS Tagger (MXPOST). An interface was created for the XTAG morphological analysis dictionary [6] to enable retrieval of morphological information for factored processing. It was observed that different tagsets are used by the various POS taggers, the Moses system, and the Penn treebank. A tagset conversion algorithm was designed to allow better comparison across data sets and to allow conversion between tagsets.

Urdu: Two methods for Urdu POS tagging were compared: The LCTL-supplied Urdu POS tagger and POS lookup using the LCTL Urdu lexicon [4]. Lexical look-up was easier to use, but was restricted based on the limited POS information in the lexicon. In order to use the LCTL tagger, programs were written to put each word of a sentence on a separate line and add a dummy tag (“*KPOSPrep.java*”) and then to reconstitute the sentence after tagging (“*ldcpstag2moses.pl*”). For lexicon look-up, a program was written to access the LCTL lexicon and return the stored POS data for each word (“*UrduGetPOSonly.java*”).

An analysis was also made of the effect of spelling normalization on the POS tagger. Normalization of Arabic spelling to Urdu spelling improved the performance of the tagger, by making the input more similar to the data used to train the tagger.

Russian: The Treetagger program was evaluated for Russian POS tagging. Treetagger returns the POS and the lemma (which is the canonical form, as opposed to the simple stem). For Russian, a larger tagset is also available that contains more detailed morphological analysis.

Stemming: Inflection causes problems for MT when a single word has multiple inflected forms; this reduces the ability of the system to match patterns across languages. Stemming converts all inflected forms of a word to a single base form.

Urdu: Added POS Restrictions: Development continued on a list-based Urdu stemmer. The stemmer was improved with POS restrictions, so nominal endings are only removed from nouns, verbal endings are only removed from verbs, etc.

Russian: Snowball Stemmer: An analysis was conducted on the effect of applying the Russian Snowball stemmer to MT in Joshua. Stemming reduces the number of OOV words (since the training data may contain a version of the word with a different ending). Stemming with the Snowball stemmer improves the scores for Joshua (see Section 2.1.15).

However, stemming also appears to cause problems in the translation of pronouns, since the forms of *on oha oho ohu* = *he she it they* become identical after stemming. Increasing the minimum stem length could improve this output.

French: Problem in Tokenization in Snowball Stemmer: A review of the results of the French Snowball stemmer pointed out that the stemmer is designed to replace some accented vowels with unaccented vowels in the stem, even in non-inflected forms. For example, in the following sentences the accent mark is removed in both the stemmed word, *tolèrerait*, and in the unchanged stem, *très*.

Original	Ca peut être <u>très</u> compliqué , l' océan .
Stemmed	Ca peut être <u>tres</u> compliqu , l' océan .
Original	Pas <u>très</u> souvent . On ne le <u>tolèrerait</u> pas .
Stemmed	Pas <u>tres</u> souvent . On ne le <u>toler</u> pas .

This may not be the desired behavior for SCREAM Lab projects since it creates a separate word *tres*, distinct from the original word, *très*.

Splitting and Compounding Normalization for Urdu: Rule-based and frequency-based splitting programs were created to standardize Urdu compound words.

Types of Variation: Urdu differs from English in its expression of acronyms, the use of spaces to indicate word boundaries, the segmentation of borrowed words, and the attachment of postpositions.

Acronyms as Phrases: Translated acronyms like BBC are generally written out in Urdu with one word per letter *بی بی سی* /by by sy/. This creates a problem for word alignment; there can also be confusion when the elements of the acronym coincide with existing Urdu words. For example, a word-by-word translation of “by by sy” is “teal teal thirty.”

Two acronyms were noted in the data that follow the English pattern of one letter per word: the Pakistani agencies NEPRA (National Electric Power Regulatory Authority) and WAPDA (Water and Power Development Authority). These are written as single Urdu words: /nyprA/ نیپرا and /wApdA/ واپڈا.

Non-Joiners Allow Writers to Omit Spaces: Arabic script languages often appear with word boundary errors that can impede named entity detection. Some characters are linked, but others are non-joiners, giving the appearance of a space within a word. Sometimes writers fail to type an actual space between words when there are non-joining characters; alternatively, writers may add spaces within words. For example, the Urdu data include the sequence اور برازیل /Awr brAzyl/ which is the run-together form of اور plus برازیل /Awr brAzyl/ "and Brazil." Since the letter /r/ does not connect to the following letter, the run-together form preserves the appearance of word spacing.

Borrowed Words as Phrases: Some borrowed English words are written as multiple words in Urdu, leading to the creation of extra words. For example, Urdu borrows the English word *association* but spells it as two words, ایسوسی ایشن /Ayswsy Ayšn/ creating an apparent word, *ation* (See Section 2.1.11).

Postpositions: Urdu marks grammatical relations with postpositions, which are similar to English prepositions but follow rather than precede the noun phrase. Postpositions in Urdu may stand alone as separate words, but often they are written attached to the preceding element, similar to case inflections in fusional (e.g. Latin) languages. Conjunctions are also sometimes attached to the preceding element. For MT, we prefer to normalize the data by separating all conjunctions and postpositions if possible.

Rule-based splitting to remove all potential conjunctions and postpositions corrects many of the run-together words like برازیل /Awr brAzyl/ "and Brazil," but also has the potential to split up legitimate unknown words like San Francisco سین فرانسیسکو /syn frAnsksk/ and Orlando اورلینڈو /Awrlɪndw/ (taking *-ko* /kw/ as a postposition, and *aor-* /Awr/ as the conjunction, respectively).

A program was written, "*KFrequencySplit.java*" that splits off conjunctions and prepositions in accordance with word frequency: The word is split if the geometric mean of the frequencies of the parts is greater than the frequency of the original word. This prevents splitting when the remaining sequence is not a possible word.

Bigram/Unigram Splitting: Previous work addressed Urdu compounding by using a dictionary list to detect possible split and compounded words, such as ایسوسی ایشن /Ayswsy Ayšn/ "association" and جوبائیڈن /jwbA'ydn/ "JoeBiden"; corpus frequency counts were then used to determine which was more desirable, the two-word or one-word variant.

In this effort, the decision was made to implement a strictly corpus-based spelling correction, which identifies unigrams that have corresponding bigrams in the same data set and normalizes to the more frequent option. At a practical level, the goal is not to create correct spellings, but rather to group variants together appropriately; this creates more examples of each word and gives the MT a better chance to match each word to its translation.

An algorithm was designed that compares the bigram and unigram counts for the same phrase, converting the less frequent version to the more frequent one. For example, if the data contain both the bigram *snow fall* and the more frequent unigram *snowfall*, the program will convert all instances of *snow fall* to *snowfall*. The algorithm was implemented with a set of programs that record the bigrams and unigrams in a dataset (using the existing programs "*text2wfreq*," "*text2wngram*," and "*sort_ngrams.pl*"), generate the mapping rules ("*KCondenseBigramFile.java*," "*KGenerateBigramRules.java*"), and apply the mapping rules ("*KSpellChangeThreshold.java*").

This bigram/unigram splitting technique appears effective in splitting up compounded conjunctions and postpositions in the Urdu data, as well as removing extra spaces in misspelled words; it also corrects many examples of run-together words in the English data.

Testing of the bigram-splitting program showed that a frequency threshold was useful, allowing frequently occurring elements to remain unchanged. This improves translation for two reasons: 1. If a phrase like “associ” “ation” occurs frequently, the MT can generally handle it as is, because it sees enough examples of the matching Urdu and English words. 2. Some bigram-unigram matches represent distinct phrases, rather than errors; such legitimate phrases are generally more frequent than errors. For example, both *Pakistan is* and *Pakistanis* occur with high frequency in the English data; these are both legitimate phrases, and a frequency threshold prevents the program from trying to correct them.

Bigram splitting was tested in conjunction with digit-to-word conversion, and it was determined that splitting should be applied before digit conversion, to prevent derived numeral words from being considered part of a bigram.

A preliminary trigram-unigram splitting and compounding program was also developed, based on the bigram-unigram splitting program.

Digit-to-Word Conversion: Another source of word-level variation is the alternation between digits (1 2 3) and numeral words (*one two three*). Development continued on existing programs that convert digits to words in Chinese, Urdu, and English.

Chinese: Fractions, Percentages: The Chinese digit conversion program was extended to handle fractions and percentages.

Urdu: Phone Number Patterns and Time Expressions: A native speaker consultant provided an analysis of Urdu phone number patterns, in which large numbers are re-grouped to avoid leading zeroes. The consultant’s algorithm was adapted for inclusion in the Urdu digit-to-word conversion program.

An analysis was made of variation in time expressions within the Urdu/English parallel data. This included variation in the English spellings (*am* or *a.m.*, 7:00 or 7.00), as well as a comparison of English time expressions with Urdu time expressions (*a.m.* and *p.m.* vs. صبح *subah* 'morning,' دوپہر *do pahr* 'early afternoon,' سہ پہر *se pahr* 'late afternoon,' شام *sham* 'evening,' رات *rat* 'night').

Urdu Named Entity Tagging: The named entity tagging program was modified to replace Urdu words with the tag representing the type of named entity, such as ORG for “organization.” This may allow the MT system to learn general patterns for named entities.

Russian Morphological Characteristics: A review was made of existing literature on Russian MT, including dependency parsing, word nets, morphological analysis, and variable word order.

Initial MT output was examined, and it was noted that certain characteristics of Russian grammar can cause problems. These include sentences without subjects (a problem for word alignment), relatively free word order, and the use of gendered pronouns for inanimate nouns.

Pronouns create a translation problem for the English/Russian language pair, since English uses the neuter pronoun *it* to refer to inanimate nouns, while Russian inanimate nouns may be masculine, feminine, or neuter. Taking a count of just the lowercase instances shows the variation in distribution of the pronouns:

<u>English Pronouns</u>		<u>Russian Pronouns</u>	
he	61	он	101
she	2	она	33
It	329	оно	9

For example, in the following sentence, the word *water* corresponds to the pronoun *it* in English, but in Russian, the antecedent вода "water" is feminine and requires the feminine pronoun, она:

And when it works its way through the system and leaves, the water is cleaner than when it entered.

Но когда вода проходит через всю систему и листья, она становится чище.
literally: but when water goes-through through all system and leaves, she stands cleaner.

2.1.13. French-to-English MT Output

The IWSLT French-to-English MT output showed significant variance in scores across years, as shown here.

<u>MT System</u>	<u>2011</u>	<u>2012</u>	<u>2013</u>
2011	35.28	--	--
2012	31.43	32.93	--
2013	39.35	39.76	37.05

A review was made of the data and the translation systems in each year to look for differences that might explain the difference in scores. Various small distinctions were discovered, but the overall reason for the difference remains unexplained.

Punctuation: Certain French punctuation patterns can affect the performance of tokenization and detokenization. French punctuation traditionally uses angled quotation marks called guillemets: « » Traditional French usage also requires a space before the set of punctuation marks, ?!;#\$ and %, as well as a space between the guillemets and the quoted material. These spaces are removed by the detokenization program.

The 2011 and 2012 test data differ in their treatment of punctuation, with the 2012 data tending to follow the traditional guidelines. The training data has mixed usage.

	<u>«</u>	<u>»</u>	<u>Space Before ?</u>	<u>No Space Before ?</u>
2011	10	67	13	31
2012	56	5	57	3

Similar numbers of apostrophes were found in the 2011 and 2012 French references, suggesting that there is not a significant difference in the number of contractions. Hyphenation was examined as a possible source of problems for tokenization, since the SMT2 tokenization process removes all hyphens from the French data. Hyphens are required in some lexical items (*est-ce*

que, “is it?”) and in the inverted structures used to form questions (*avez-vous* “do you have?”). However, there were only small numbers of such hyphens in the reference files.

Vocabulary: OOV words were also examined. A program, “*CountNovelWords.java*,” was written to compare output words to the set of words in the training data. There were slightly more OOV words in the 2011 files, as well as more unique words overall. There appear to be more named entities in 2011, and hence more uppercased words. These differences seem minor and not likely to explain the difference in scores.

Sentence Alignment: The datasets were reviewed for problems in sentence alignment, such as sentences that run across multiple lines, repeated sentences or phrases, or lines that contain multiple sentences. There was no evidence of repeated phrases or sentences that take more than one line, but there were several lines containing multiple sentences, as indicated by the presence of additional punctuation. An existing program, “*KGetExamplesInverse.java*,” was adapted to identify these sentences.

The test data for 2012 have more lines with multiple sentences, and these are longer sentences. While the 2011 data have some lines with multiple sentences, these tend to be short. Here, the sequence of period plus following space is used as a rough indication of a line that may contain multiple sentences.

<u>Year</u>	<u>Lines</u>	<u>Words per Line</u>
2011	15	15.33
2012	58	29.61

The test data for 2012 also have longer lines overall. All of these factors might make the 2012 data more difficult to process (see words per line below).

<u>Year</u>	<u>French</u>	<u>English</u>
2011	16.02	15.16
2012	17.80	16.50

Contractions and Truecasing: The process of expanding and later restoring contracted words can sometimes hurt BLEU scores, as discussed in Section 2.1.9. When the word choices match the target, recontraction can help, but when the word choices are wrong, recontraction may make things worse. For example, recontraction restores a match when applied to *le écosystème* to generate the desired form *l'écosystème*, but recontraction removes a match when applied to *le éditeur* to generate *l'éditeur*, when the target is the two-word phrase, *le rédacteur*.

This may affect the 2011 and 2012 data differently. Looking at just words beginning with the accented vowel *é*, it seems that recontraction had a more beneficial effect in 2011 than in 2012. For 2011, recontraction resulted in seven better matches and three worse; for 2012, there were eight better and seven worse. A more detailed investigation of contraction is needed.

Truecasing, as mentioned in Section 2.1.9., can either help or hurt the BLEU scores, depending on whether the correct word has been chosen for the initial word of the sentence. Truecasing errors were compared across years, with similar results for the 2011 and 2012 data.

	<u>Lines</u>	<u>Words with Wrong Case</u>	<u>Average Wrong Words per Line</u>
2011 Truecased	818	313	0.278
2012 Truecased	1124	205	0.251

2.1.14. Transliteration

Transliteration is the mapping of letters to letters across languages, and can be useful in MT for borrowed words that would otherwise remain untranslated.

Script-Based Transliteration for Language Conversion: Some related languages such as Hindi and Urdu differ primarily in their writing systems. Systematic transliteration from one writing system to the other can potentially convert data in one language into the other language. Preliminary script-based transliterators were developed for Hindi/Urdu and Farsi/Tajik, converting from Devanagari to Arabic script, and from Arabic script to Cyrillic, respectively. A word frequency list is used to select among transliteration options. For Tajik and Farsi, word frequency lists were built from Wikipedia articles.

Difficulties were found in transliterating between Hindi and Urdu, because the mapping between letters is many-to-one. For example, Urdu has four characters that represent [z], while Hindi has one. Further work would also be needed to map Hindi conjunct consonants and diacritics used for foreign sounds. Later work under Task Order 14 [7] reviewed existing Hindi/Urdu transliteration programs and research, and noted that vocabulary choices between Hindi (which draws on Sanskrit) and Urdu (which draws on Persian and Arabic) make it difficult to generate parallel text via transliteration.

Transliteration of Borrowed Words: Borrowed words that would otherwise remain untranslated may be addressed with transliteration. Transliterations from foreign names or other words that are sounded out into English (e.g. *Muhammed*, *Mohammad*, etc., from Urdu محمد /mhmd/) may generate a lot of spelling variation, and these transliterations are typically handled in MT with statistical pattern matching. This effort focused instead on back transliterations, which are words borrowed from English into the other language (e.g., Urdu زیرو /zyrw/ , borrowed from *zero*). Back transliterations from Urdu to English have fairly consistent sound correspondences, and may be addressed with rule-based sound mappings.

Borrowed Words in Urdu: Borrowed words are particularly important in Urdu to English MT, because of the prevalence of borrowing and because of ambiguities between borrowed words and native Urdu words.

Prevalance of Borrowing in Urdu: Urdu writers may borrow from English even when native words are available. The borrowed word زیرو /zyrw/ “zero” co-exists with the Urdu word صفر /Sfr/ [sifar] “zero.” Whole sequences of words may be borrowed, as shown here. The examples are given in three lines: English, Urdu, and Urdu sounds with transliterated words underlined.

In the top ten, India comes in the last

اس سرٹیفکیشن کی ٹاپ ٹین میں بھارت آخری نمبر پر ہیں۔

as certification ky top ten myn bhart Ajry number pr byn

The chief operating officer of Intel movie Mr., Philip Moorish has said that these photographers

have made new standards taking best photographs from digital cameras of Intel.

چیف آپریٹنگ آفیسر فلپ مورش نے کہا ہے کہ ان فوٹو گرافروں نے انٹیل کے ڈیجیٹل کیمرے سے انٹیل مووی کے

بہترین فوٹو گرافی کرتے ہوئے نئے معیار قائم کئے ہیں۔

Intel movie ke chief operating officer Philip Moorish ne kha he kh an photo graph-on ne Intel ke digital camera-e se bhtryn photo graphy krte hoie nie mayar qaim kie hyn

Difficulty in identifying Borrowed Words in Urdu: The use of retroflex consonants may help identify borrowings: English alveolar d, t are further back than dental Urdu d, t, and so are usually written in Urdu with retroflex t ٹ and d ڈ. For example, “top ten” in the previous section is written with retroflex t: ٹاپ ٹین /t̪p̪ t̪ɪn/.

Borrowing may create homophones with existing words, such as English *Bill* = Urdu بل “twist,” English *C* = Urdu سی “thirty”; English *K* = Urdu postposition ك /ke/; etc.

An additional complication when identifying borrowed words is the use of blended morphology: Urdu speakers may borrow an English word, and then apply an Urdu inflection. For example, the dataset contains many examples of plurals with the suffix /-wn/ as in لیڈروں /lyḍrwn/ “leaders.”

Consonant-Based Transliteration: Detailed information on the Urdu to English transliteration program described here is provided in Appendix B., “Consonant-Qualified Transliteration: An Adaptable Rule-Based System for Back-transliteration of English Words in Arabic Script Languages.”

Vowel Mapping in Arabic-Script Languages: When working with Arabic-script languages, the transliteration of vowels becomes a problem. Most vowels are not written in Arabic; diacritic symbols for the vowels exist but are only used for pedagogical or religious purposes. In Urdu, there are symbols for the long vowels, but the lax vowels are typically omitted.

Reviewing the Design of the LCTL Urdu-to-English Transliterator: The LCTL Urdu-to-English transliterator was evaluated. The LCTL transliterator maps the Urdu symbols to a consonantal skeleton, and then creates all the possible variants by permuting the vowel spellings. These potential words are then matched against an English word frequency list, using the English spellings.

Consonant-Qualified Transliteration from Urdu to English: An alternative transliteration method was designed, in which the consonantal skeleton derived from the Urdu symbols is used to select potential sound sequences of actual English words. This was termed consonant-qualified transliteration, since only words which match the consonants are qualified for further consideration. Separating the consonant-matching and vowel-matching allows us to reflect a higher level of confidence in the consonantal matching.

A new transliterator program, “*CQTranslit.java*,” was created that improves performance with words that have been borrowed from English into Urdu. The new program follows the rule-and-dictionary approach of the LCTL transliterator, but first maps the Urdu characters to sounds, and uses the CMU English pronunciation dictionary instead of a word frequency list in English spelling. As in the LCTL transliterator, this program first matches consonants, and then deals with vowels, but instead of considering every vowel permutation between every consonant pair, “*CQTranslit.java*” uses a language-specific mapping. The Urdu vowel characters are mapped to ipa sounds; the program is enabled to skip lax vowels in the English entries, since these are often not written in Urdu. The search space is reduced by only considering English words that match the consonantal pattern during the vowel mapping phase, making the program more efficient.

Basic Design: The basic method maps Urdu symbols to an intermediate, phonetic representation, which is then compared against a phonetic English word list.

First, the Urdu symbols are mapped to an intermediate, phonetic representation. Urdu consonant symbols are mapped to consonant phonemes; Urdu vowel symbols are mapped to a set of possible vowel sounds. Then, the sound sequences are matched against a list of English words in phonetic representation. The English word list was created from the Carnegie Mellon University (CMU) English pronunciation dictionary, annotated with word frequency counts derived from the training data. Weights determine the choice among the best matching and most frequent potential English words.

When comparing sound sequences to the wordlist, the English lax vowels [ə ɪ ɛ ʊ] do not have to be matched, since these sounds may be omitted in Urdu writing. The Urdu vowels that are present are matched according to various options, as indicated in Section 2.1.14.

Vowel Mapping Methods: The Urdu non-lax vowels are handled by one of three vowel-mapping methods. In the strict consonant-and-vowel matching condition (CV), each vowel must be mapped to an appropriate phoneme in the English word. In the consonantal matching condition (C), a placeholder is inserted for each Urdu vowel symbol, and this placeholder vowel is allowed to match any vowel sound in the candidate English word.

An evaluation was conducted using the program Scrite to score word and character error rates. The CV transliteration program outperforms the C transliteration program in those cases in which the CV program is able to make a guess, but the CV program often fails to find an interpretation of longer words. A variation in vowel spelling may block an otherwise reasonable CV candidate. The CV version also requires more time to evaluate the transliterated words.

A third vowel mapping method was therefore introduced, which first identifies the consonantal matches, then searches these consonant-qualified (CQ) words for full consonant-and-vowel matches. If no match is found the program then backs off to the best consonantal match. Using CQ transliteration solves the problem of excessive search time for the vocalic phase, and lets the program take advantage of both high-confidence consonant information and lower-confidence vowel information. Testing shows that the CQ method outperforms either consonant or consonant-and-vowel matching.

A comparison showed that a statistical transliteration program developed by the SCREAM Lab partners at MIT/LL has advantages for processing Arabic and Urdu named entities, while the rule-based CQ transliterator is better at transliterating words that are borrowed from common English nouns.

Dictionary Look-Up Methods: An examination of the words that remain unknown when transliterating with the CMU pronunciation dictionary for American English identified these problem areas: abbreviations, acronyms, misspelled words, British English variants, and Pakistani English vocabulary. The transliteration program was adapted to allow multiple dictionaries, so the CMU dictionary can be combined with domain-specific dictionaries for these areas, using existing SCREAM Lab text-to-speech software to generate the pronunciation entries.

LMs: An attempt was made to improve the transliterator by adding contextual information to guide word choices. For example, one of the borrowed words is the name *Abraham* in *Abraham Lincoln*, spelled in Urdu as ابراہم لنکن /AbrAhm lnkn/. The consonantal transliterator identifies

several words that match the consonant pattern, including *Abraham* and *Ibrahim*; of these, *Ibrahim* has the highest word frequency in the data, and is therefore selected as the most likely transliteration. The hypothesis is that including information about the surrounding words (i.e., accessing statistics from a LM) will allow the program to make more reasonable choices. The program was revised to allow weighting of the word frequency score against the LM score, so the amount of emphasis on context can be varied. This did not provide the expected benefit.

Two problems with this approach are (1) a word which is uncommon in word frequency counts may also be uncommon compared to other words in contextual statistics, and (2) the transliterator is being applied at the end of the MT process, after computer-generated word re-ordering and word choice, so the contextual information may have been disrupted. In the *Abraham Lincoln* example, the MT process had already attempted to translate the word لنکن /lnkn/, and erroneously generated *Lankan* (as in *Sri Lankan*). This takes away the context, *Lincoln*, which would have presumably helped select *Abraham*.

Stemming: A variation of the program was created that combines transliteration of borrowed words with stemming and translation of Urdu inflectional endings, to handle words like لیڈروں /lyḍrwn/ *leaders*, which contains the borrowed English word *leader* with the Urdu plural ending, /wn/. Variant forms of the word are created in which the Urdu ending is replaced with the English plural options [s], [z], or [əz]; the correct word can then be identified in the English pronunciation dictionary.

Application of Consonant-Qualified Transliteration to Other Arabic-Script Languages: Specific letter-to-sound mappings were created to adapt the program to Arabic, Dari, and Pashto. For example, the character for Urdu [b] can represent either [b] or [p] in Arabic, so this mapping has to be added. Test sets of borrowed English words in Arabic and Dari were created by looking up place names in the new SCREAM Wikipedia Aided Translation (SWAT) database.

Some consideration was given to the possibility of creating a generic Arabic-script transliterator. The original Urdu-to-English transliterator was tested without modification against Dari text. The program successfully identified some of the borrowed words in a news article, including “September,” “UNICEF,” “polio” and “vaccine;” other borrowed words were not correctly identified. The Arabic transliterator with expanded letter mappings was tested again on Urdu, and was found to generate a small number of errors with words in which the new mapping allowed f/v and p/b alternations not found in Urdu.

Application of Consonant-Qualified Transliteration to Russian: The consonant-qualified transliteration method was extended to Russian, and demonstrated a basic capability to transliterate borrowed English words found in Russian text. Unlike the Arabic script languages, Russian does not omit vowel symbols; however, a single Russian vowel symbol can represent several sounds, depending on stress patterns. Prioritizing for consonantal mapping is therefore still helpful. Details on Russian transliteration are found in Section 2.1.15.

2.1.15. OOV Word Handling

Words that are not found in the training data may persist untranslated in MT output. Finding and using translations for these OOV words may improve the translation output. These OOV translations may be derived by various methods such as dictionary look-up, by transliteration (for names or borrowed words), or by analysis of inflected forms.

When the language pair involves different scripts, OOV words may be easily identified in the output of MT. For language pairs like English/French, however, the OOV words are not always obvious. A method was developed to identify the words of the input file that will remain untranslated via inspection of the phrase table entries.

Discussion with the government program manager identified three ways to use translated OOV words within the MT system:

1. Run the MT, identify and translate the OOV words in the output file to create additional output sentences, and then rescore the output sentences to select the best ones.
2. Identify OOV words and record translations for them in a mapping, use that mapping to pre-translate the OOV words in the input file, then run the MT
3. Identify OOV words and record translations for them, add these mappings to the phrase table that guides translation, and then run the MT.

Working with the Russian translations, a fourth option for applying the OOV translations was developed:

4. Identify the OOV words, record translations of these words as lattice options in the input file, and then run the MT.

Russian: Variants for Inflected or Transliterated Words: An inspection of Russian OOV words reveals two main sources: inflected words and borrowed words. Many of these can be recovered for translation by considering other inflectional endings, and by transliterating the borrowed words.

Sources of OOV Words: Russian-to-English MT exhibits a high proportion of OOV words. An analysis of OOV words when translating via the Joshua MT system detected two sources of untranslated words: borrowed words or names, and inflected forms of known words. Borrowed words and named entities are domain-dependent, and hence may not be found in the training data (for example, an article on video gaming included the previously unencountered words геймер /gejmer/ “gamer” and варкрафт /varkraft/ “Warcraft”).

Another main source of OOV words is the inflection of Russian nouns, verbs, and adjectives. A noun may occur in the training data in a variety of forms (singular, plural, nominative, genitive, instrumental, etc.), but if it appears in the test data with a different inflectional ending, it will remain untranslated.

It was also observed that a word may be found in its exact form in the training data, but still not translated in the test data, presumably because of limitations in word alignment and phrase extraction.

Stemming to Recover Inflected OOV Words: One approach to the presence of unknown inflected forms is to remove the inflectional endings. This groups all the inflected forms of a word under a single stemmed form, reducing data sparsity.

Stemming All Inflected Words: A traditional approach to the MT of highly inflectional languages is to stem all of the word forms in the training and test files. Stemming removes inflectional endings and reduces data sparsity. The Snowball stemmer for Russian was therefore applied within Joshua and was found to improve the score by about one-half point. The number of unique OOV words decreased from 1354 to 451.

Stemming with the Snowball stemmer improves the scores for Joshua (results here shown for both Moses tokenization and Joshua’s default Penn treebank tokenization):

	<u>Moses Tokenized</u>	<u>Penn Treebank Tokenized</u>
Baseline	15.67	15.27
Stemmed (Snowball)	16.10	15.51

The Snowball stemmer did not improve results as much for the Moses MT system, perhaps because the Moses system has fewer initial OOV words.

Frequency-Based Stemming: Stemming reduces data sparsity but also removes potentially useful information. An attempt was made to preserve some of that information, by restricting stemming to unseen or very infrequent words.

Frequency-Based Stemming with Word List: A frequency-based stemming program, “*StemByFreqN.java*,” was written to stem words according to their frequency in the training data. A frequency list was derived from the training data, and used to identify novel words in the test data before MT. The novel words were subjected to stemming if there was a corresponding shorter sequence available in the frequency list (in English, for example, the word *runs* could be shortened to *run*). The program also allows the substitution of a final character, to allow for inflectional alternations between a less common form and a more common form (such as *alumna* > *alumni*). The parameter N can be set higher than 1, to allow stemming of infrequently occurring words; however, the value of 1 was found to be the most useful, restricting stemming to words that never occur in the training data.

This method permits the detection and stemming of inflected forms without explicitly specifying the possible suffixes; the algorithm is therefore language-independent, and could be applied to new languages for which little morphological information is available.

Testing showed that using the rule-based Snowball stemmer for Russian was more effective than the “*StemByFreqN.java*” program. The “*StemByFreqN*” program can derive some false positives, in which an unknown word is stemmed to an accidentally similar, shorter word. In some cases, a verb form is stemmed to a related noun form, providing some semantic information but not matching the actual translation. For example, the verb *исходила* “was coming from (feminine singular)” was not found in the training data; “*StemByFreqN*” stemmed it to the related noun *исход* “outcome.”

Frequency Stemming with Phrase Table Filter: Some OOV words are found in the training data but fail to be extracted to the phrase table. These words were originally protected from stemming. A variant of the “*stemByFreqN.pl*” program was created, introducing a condition, “*PTfilter*,” that checks for the presence of the candidate word in the phrase table.

The results of the various stemming programs were measured in two ways: by the number of OOV words remaining in the output, and by the overall score. Both “*stemByFreqN*” and the Snowball stemmer reduce the number of OOV words; a combination of these generates the smallest number of OOV words:

	<u>Original</u>	<u>StemByFreqN</u>	<u>StemByFreqN with PTfilter</u>
Baseline	1354	993	839
Stemmed (Snowball)	451	317	

However, the scores achieved using *stemByFreqN* do not improve over the baseline:

	<u>Original</u>	<u>StemByFreqN</u>	<u>StemByFreqN with PTfilter</u>
Baseline	15.67	15.41	15.43
Stemmed (Snowball)	16.10	15.76	

A qualitative analysis of a dozen lines of the test data shows that the phrase table filtered option does improve the treatment of formerly OOV words. Four words are correctly translated and several other words are translated as related words:

<u>Reference</u>	<u>Output</u>	<u>Changes to the Russian Words</u>
passage	passage	correctly stemmed
shock	shock	correctly stemmed
times	times	corrected misspelled final character
lopez	lopez	corrected misspelled final character
symphonies	symphony	plural ending removed
instrumental	tool	adjectival ending removed
black	the blacks	adjectival ending removed
was coming from	outcome	verbal ending removed
nathaniel	nathan	truncated name
throws in the towel	refuses	semantically correct

This suggests that there may be semantic value to using stem-by-frequency methods on OOV words.

Stem and Inflect: The stem-by-frequency method is agnostic with respect to particular language morphology. In contrast, the stem-and-inflect method leverages knowledge of the inflectional system of the language under study, reducing the variety of forms that may be considered.

Stem and Inflect Method: At the suggestion of the government program manager, an improved process for identifying variant inflected forms was developed, which stems and then re-inflects the OOV words according to the rules of the Russian inflectional system, generating all possible forms of the word.

For example, in the output for the test sentence below, the OOV words are аквакультура “aquaculture” and рыбное “fish,” both modifying the word for “farming:”

... but аквакультура рыбное farming , is going to be a part of our future .

The meaning of the word рыбное “fish” can be determined by looking at its related forms, рыбная,рыбного,рыбное,рыбной,рыбном,рыбному,рыбную,etc. Some of these variant forms are present in the training data, and thus have phrase table entries that can be used to translate the original word.

<u>Source</u>	<u>Word</u>	<u>Phrase Table Entries</u>
Original	рыбное	none (OOV)
Variant	рыбного	Fish story
Variant	рыбной	Of
Variant	рыбную	equivalent lost fishing license
Variant	рыбные	fish
Variant	рыбный	fish fishing land and the fish
Variant	рыбных	of the fish of we

The stem-and-inflect algorithm was initially designed with the following steps:

1. Identify the OOV words by looking at the baseline MT output 1-best list, using the program *“findCyrillicWords.pl.”*
2. Get the part of speech for each word using the existing Treetagger program.
3. Stem the word using the revised RevP stemmer, *“RussianStemmer2013.java”* (see Section 2.1.18).
4. Create all possible inflected forms for that stem and part of speech, using a program derived from the RevP stemmer, *“RussianInflectionGenerator.java.”*
5. Look for any unigram instances of these inflected forms in the phrase table and collect the English translations, using the program *“CollectTranslationsFromPT.pl.”*
6. Augment the baseline MT output N-Best list by alternating the OOV words with the new translations, using the program *“RingChanges.java.”*
7. Use Joshua to extract a new 1-best list from these alternatives.

Originally, the Snowball stemmer was applied in Step 3; however, this generated errors when adding the inflectional endings, because the Snowball program provides lemmas, instead of clipped stems. For example, for verbs, Snowball generates the infinitive form, which consists of the verb stem plus the infinitive suffix, -ть. This infinitive ending needs to be removed before the inflectional endings can be added. Step 3 was modified to use instead the stemmer developed for the Reverse Palladius (RevP) program (See Section 2.1.18). The RevP stemmer is a single-pass, greedy stemmer that removes the longest potential inflectional ending leaving the clipped stem form. The RevP stemmer was extended to include verb inflections and special handling for the verbs of motion, which use different stems in the present and past tense.

In addition, three optional variants were added to this process:

- a. Restricting the collected translations to exclude uninformative unigrams such as “the,” “a” and “of” that can result from partial alignment to a noun phrase.
- b. Splitting hyphenated OOV words and processing each part separately (See Section 2.1.15).
- c. Transliterating the remaining untranslated OOV words (See Section 2.1.15).

Various options were tested. The stem-and-inflect process improved the translation qualitatively but did not improve the translation scores.

<u>Joshua System</u>	<u>MultiBLEU (Official Scoring)</u>
Baseline (best of 10 optimization runs)	15.67
stem-and-inflect OOV	16.64
stem-and-inflect OOV with hyphen splitting	15.65
stem-and-inflect OOV with hyphen splitting and subsequent transliteration	15.69

The number of words remaining in Cyrillic was almost cut in half, going from 1354 to 700 words; however, not all of these represent correct translations. In addition, the scoring script assesses a penalty for longer sentences, so the process can hurt the score if it introduces additional words. For example, if the program selects the entry [рыбных = of the fish] to translate рыбное “fish,” it adds two unwanted words.

Augmented N-Best Lists vs Input Lattices: The stem-and-inflect OOV process described above creates extremely large augmented N-Best lists, especially when hyphenated words are split. Input lattices were created as an alternative for recording the variants of the Russian OOV words. The OOV words are stemmed and inflected as before; any forms that are present in the phrase table are entered as alternatives to the original word in a word lattice of the original Russian sentence. A transliterated version of the OOV word is also included in the lattice. The lattice versions of the input sentences allow Joshua to access the various possible translations during the MT.

The OOV word рыбное “fish” used as an example in Section 2.1.15 generates this lattice structure for the original sentence:

но аквакультура, рыбное|рыбного|рыбной|рыбную|рыбные|рыбный|рыбных
фермерство, будет частью нашего будущего.

From this, Joshua can consider alternate input sentences and see which one gives the best translation; the intention is that Joshua will make use of one of the known alternatives to рыбное, and generate a translation with the word “fish” in the output.

The Joshua system appears to have trouble processing lattice input, at the step in which Joshua creates a filtered phrase table based on the words that are present in the input. Research shifted to the Moses system, which seems to have better handling of lattice input. The collectUsefulVariants program was revised to read from a Moses-formatted phrase table.

As with the N-Best lists, the stem-and-inflect lattice method reduces the number of OOV words by about half, with most of the new translations being semantically meaningful, and many of the translations matching the target, leading to improved BLEU scores.

Some of the new translations included the target word but added additional, unwanted words. Initially, the lattice weights for the inflected variants were made proportional to the frequency of those words in the training data; these were later set to equal weights for all the variants found in the phrase table, in hopes that the system would select arcs with shorter translation phrases. Equal weighting provided the best output, with BLEU scores improving from a baseline mean of 0.1372 to a mean of 0.1588.

Hyphen Splitting: Hyphenated OOV words sometimes consist of known parts. The “findCyrillicWords” program was revised to collect and annotate the separate parts of hyphenated OOV words, and a new program, “checkPTforHyphenatedForms.pl,” was written which retains the separate pieces only if they are not found in the phrase table. The rationale here is that, if the pieces are known, simply splitting the hyphenated word will enable their translation. For example, the OOV бизнес-школ “business school” can be resolved by splitting it into бизнес – школ, since each word is found separately in the phrase table. If, on the other hand, the pieces are not known, they are retained on the OOV list for additional processing.

Transliteration to Recover OOV Words that are Borrowed Words or Named Entities:

Transliteration was applied to OOV words that could not be found through the stem-and-inflect process, on the assumption that these might be borrowed words or names. A program developed for Urdu transliteration, “CQTranslit.java” program, was revised to apply to Russian, and renamed “TranslitCyrillic.java.” The “CQTranslit.java” program first matches the consonantal skeleton of a word, and then weights various vocalic options within that form, based on frequency data (See Section 2.1.14).

The “*TranslitCyrillic.java*” version maps Cyrillic characters to their typical sounds, and also allows for alternations such as /v/ for /w/ and /t/ for /θ/ in borrowed words. The sound mappings are first compared consonantly, and a version of the CMU English pronunciation dictionary is used to detect possible English words. Another program, “*ReweightDictionary.java*,” was written that allows the dictionary to be reweighted according to a new training set; this was applied using the current Russian training data. The “*TranslitCyrillic.java*” program was adjusted to keep any options that matched both vowels and consonants, whether these were present or not in the training data, as well as any options that matched all the consonants while also appearing in the training data. For example, for the OOV word скид [skid] “skid,” the vowel-matching options *skied* and *skid* are retained, even though *skid* does not occur in the training data. Of the many possible consonant-matching options, only *scad* and *scud* are retained, since these are the ones that occur in the training data.

It was noted that some transliterated words need to be stemmed before they can be recognized as English words. For example, пассажа [passaža] “passage” needs to be stemmed to пассаж [passaž] before it can be properly transliterated. The transliteration program was therefore revised to create “*TranslitCyrillicAllowingStemming.java*.”

The lattice creation steps were revised to include the output of both the stem-and-inflect process and the transliteration process, in anticipation that the MT system would select the correct transliterated form as the most likely “Russian” input; this transliterated form could then pass through unchanged, since no Russian-to-English phrase table entries would match it. For example, given a lattice input of пассажа|passage, the program is expected to choose *passage*, which would then remain unchanged. Often a different transliteration is selected, however, and sometimes the original Russian form is retained instead, despite having the lattice weight for the Russian word set to zero. A further revision to the program was made that removes the original form altogether, in order to force the system to choose one of the transliterated forms.

Adding transliteration to the stem-and-inflect process yielded mixed results. In one test, candidate English forms were found for 55 words; of these, about one quarter were correct.

<u>Russian OOV Sounds</u>		<u>CQTranslit Form</u>	<u>Correct Form</u>
терминатор	[terminator]	terminator	Terminator
пивот	[pivot]	pivot	Pivot
скид	[skid]	skied	Skid
найдя	[naidja]	nadia	finding

Here, *terminator* and *pivot* are correctly transliterated, as the program abstracts away from the vowel differences ([i]=ee, [a]=ah) to derive the English words. The vowel differences prevent the correct determination of *skid*, since *skied* is a closer match to the Russian vowel. Finally, найдя is a Russian verb form that should not be transliterated; the transliteration program incorrectly selects the phonetically similar name, *nadia*.

In some cases, the transliteration component picks some unusual options, because they happen to occur in the phrase table within English-to-English entries (this can occur if the training data includes Russian sentences with some borrowed English words such as *dna*).

For example, the named entity дауни “Downey” is an OOV word that cannot be found through inflection; transliteration gives us several possibilities, including the desired target *downey*, as well as *dinah*, *denny*, and *dna*. The translation system takes the lattice input as “Russian” and

attempts to find translations in the Russian/English phrase table; normally this results in the “Russian” form being passed through untranslated. Unfortunately, the transliterated word *dna* does have a phrase table entry, dna|giving, and this translation is chosen. So, instead of *robert downey* the system creates *robert giving*.

<u>Russian OOV Sounds</u>	<u>CQTranslit Form</u>	<u>CQTranslit Form</u>	<u>Correct Form</u>
дауни	downey	dna (>> <i>giving</i>)	downey
	dinah		
	denny		

Transliteration is thus an appropriate choice for many of the remaining OOV words after the stem-and-inflect process, but it is difficult to apply in an unsupervised manner.

Interaction of Spelling Normalization and Word Alignment: Spelling normalization for mixed Latin and Cyrillic characters (see Section 2.1.9) was also applied to the Russian files used for MT with the Joshua 5 system. Such normalization did improve the score from a baseline of 15.76 MultiBLEU, to 15.9 MultiBLEU. This was not due to some OOV words being normalized and therefore matching phrase table entries, as had been expected. Inspection showed that none of the OOV words involved mixed characters. Instead, normalization of other words changed the alignments during training, causing different phrases to be extracted to the phrase table. There are some words that are OOV in the baseline condition only, and other words that are newly OOV in the normalized condition.

For example, here is one sentence that has a different OOV word in each condition. In the baseline condition, the word for *catalyst* is unknown; in the normalized condition, the word for *catalyst* is found, but the word for *invisible* is now unknown. The relevant words are underlined. Note that this sentence itself contains no mixed character words; the differences result from changed word alignment in other sentences that happen to have mixed characters.

Original Russian

Как-будто он был под воздействием какого-то невидимого лекарства, какой-то химической реакции, катализатором которой была моя музыка.

Original English

It was as if he was in the grip of some invisible pharmaceutical, a chemical reaction, for which my playing the music was its catalyst.

Baseline Output

like he was underneath the stress of some of a kind of invisible drugs , a chemical reactions , катализатором that was my music .

Output Using Normalized Training Data

like he was under by some невидимого drugs , a chemical reactions , a catalyst that was my music .

The overall OOV counts also reveal this pattern:

	<u>Total OOV</u>	<u>OOV in this Condition Only</u>
Baseline	1354	60
Normalized	1352	58

This suggests a need for continued attention to the word alignment process for the Russian-to-English MT system.

Augmented Phrase Tables: An investigation was made using augmented Russian phrase tables to derive the stem-and-inflect lattices for OOV words in the Moses dev and test data. The augmented phrase tables used were maxl5.pt and maxl5+lexapprox.pt. The lexapprox phrase table contains more entries, derived by modifying characters of OOV words to find similar entries in the phrase table; this means that there are fewer remaining OOV words for the stem and inflect process to work on.

The stem-and-inflect process was applied to find variants of the OOV words for each phrase table. The program “*idOOVs.java*” was used to look at an input file and calculate which words are not contained in the phrase table; stemming and inflection were then applied to generate possible variant forms. In order to handle asterisks in the new data, a variant of “*collectUsefulVariantsMoses.pl*” was written, “*collectUsefulVariantsMosesNoSpecialChars.pl*.” In creating the lattices, the original arc for the OOV word was removed; the new arcs for the variant forms were given equal weights, summing to 1.

<u>File</u>	<u>Using maxl5.pt</u>		<u>Using maxl5+lexapprox.pt</u>	
	<u>OOV</u>	<u>Found</u>	<u>OOV</u>	<u>Found</u>
dev2010	406	155	54	2
tst2010	551	232	65	4
tst2011	522	209	522	209
tst2012	587	253	582	249
tst2013	526	236	520	233

The hyphenated OOV words were examined to see if any of these could be translated by splitting the word into parts. Some hyphenated words were already found by inflecting the entire word.

For those that remained unknown, if at least one part of the word could be found in the phrase table independently, the split variant was added, using the program “*collectUsefulVariantsMosesNoSpecialCharsSplitHyphens.pl*.”

<u>File</u>	<u>All using max15.pt Hyphenated OOV Words</u>	<u>Found Via Stem&Inflect</u>	<u>Found Via Hyphen-Splitting</u>
dev2010	31	-	30
tst2010	38	12	22
tst2011	48	5	48
tst2012	37	5	37
tst2013	28	-	29

Arabic: Morphological Variants: An initial test was made using lattice alternatives for Arabic OOV words in the Arabic-to-English MT. The program “*findCyrillicWords.pl*” was adapted to create “*findArabicWords.pl*,” this was then used to identify the OOV words in the output file. It was determined that hyphenation is not an issue for Arabic OOV words, in contrast to the Russian data in which hyphen splitting was necessary.

A program, “*stemAndInflectArabic.pl*” was written to create additional forms of the OOV words for certain morphological alternations (such as the endings tah-marbuta 0629 ة vs. tah 062A ت). The program “*collectUsefulVariantsMosesArabic.pl*” was applied to check the phrase table for these variants. The phrase table was found to contain variants for 16 of the 57 OOV words with ة and ت. Lattices were created in the input file containing these variations, but this did not improve the overall MT scores.

An investigation was made of OOV words beginning with the prefix ال /al/, checking the phrase table to see if they can be found after /al/ has been removed. There were 121 OOV words that begin with /al/; of these, 50 can be found once /al/ is removed. This means that, altogether, 66 of the 375 Arabic OOVs can be identified by either tah-conversion or al-removal. This may not be enough to affect the overall scores, and other methods are already in use to split off the prefix /al/.

2.1.16. Chinese Word Segmentation

Chinese writing typically contains no spaces between words; in order to derive words for MT, segmentation programs are applied that break a sentence up into character groupings that maximize the creation of previously-seen words. An alternative segmentation approach is to separate each character (“character segmentation”). A lattice representation enables the MT system to maintain all these possible segmentations.

Word vs Character Segmentation: Literature on word segmentation and lattice weighting was reviewed; various existing word segmentation programs were examined, and output from different word and character segmentation programs was examined. The Stanford Chinese word segmenter was found to yield the best results. The spelling normalization program, “*ChineseNormalizer.pl*,” was updated to support better word segmentation.

Lattices vs Confusion Nets: A program was designed to create lattice representations of various Chinese word segmentations. Parentheses, apostrophes, and commas in the Chinese text were marked with an escape character, since these are used to delimit the syntax of the lattices.

The segmentation lattices were converted into HTK format, so the SRILM toolkit could be used on the lattices to generate confusion nets. Unlike lattices, confusion nets are constrained so that each possible path must pass through every node. In deriving confusion nets from the lattices, however, some ambiguities are created.

Testing of confusion nets derived from the initial, unweighted lattices reduced the translation score. An evaluation of the Chinese output identified translations in which both alternatives of a lattice sequence were included in the translation. For example, the Chinese translation of “After, we repeated this activity again with similar students” contains these sequences:

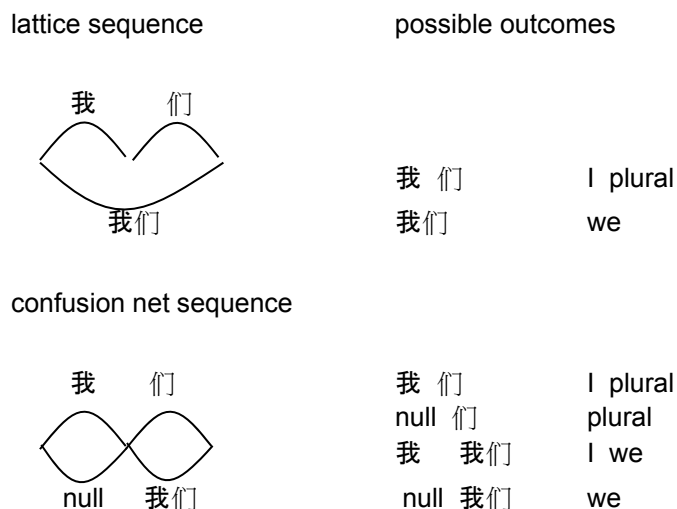
我们 I-plural “we”
 学生 learning-life “students”

The lattice representation shows that word segmentation can either choose to separate *wo* and *men* into “I” and “plural,” or keep them together as “we.” Similarly, the lattice forces a choice between treating “learning” and “life” as separate words, or leaving them together meaning “student.” The confusion net output, however, creates a path which allows both options to be taken, and the resulting translation reflects both first person singular “i” and plural “our,” and both “learning” and “live.”

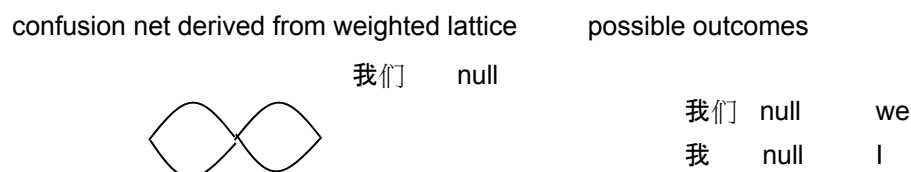
reference translation:
 After, we repeated this activity again with similar students

MT with confusion net:

i then with the same of our students learning resurrected again that live in this activity



LMs: An LM was created from previously segmented data, and the SRILM toolkit was used to assign language model probability weights to the lattice options. Initial review of the resulting confusion nets shows improvement in the treatment of alternatives.



我	们	我们	们	we	plural
我	们	I	plural		

The presence of unknown words causes errors in the confusion nets unless the language model is created with the unknown token <unk> as a substitute word. The effect of having the <unk> token in the resulting confusion nets should be investigated.

Correction to Charseg Program: An error was detected in the character segmentation program, “*cseg.pl*” that causes it to omit the first character after a string of English characters. This was corrected to create “*cseg2013.pl*.”

Example showing loss of a character after English sequence in character segmentation

(lost character: 年)

original sentence, no spaces	这是1932年在坎纳里鲁夫拍的一副照片
after cseg.pl	这是1932在坎纳里鲁夫拍的一副照片
after cseg2013.pl	这是1932年在坎纳里鲁夫拍的一副照片

2.1.17. Paraphrasing

Paraphrasing can be used to provide semantic variation in the training data for the MT system, or to expand the set of references for the optimization process. Three paraphrasing systems were explored: The pivot system of Chris Callison-Burch (CB), the Joshua/Thrax paraphrasing system, and the Paraphrase Database (PPDB).

Paraphrasing Using CB System: The paraphrasing system [8] of CB uses the parallel text from the proceedings of the European Parliament (Europarl) to find variations of English phrases. For example, given the English word *belongings*, CB’s program looks at the Europarl translations of that word into other languages, and then at all the English words associated with those translations in other sentences, locating paraphrases like *goods* and *worldly possessions*. A sentence containing the word *belongings* then serves as the basis for other sentences using the alternate words. If a parser is available, CB’s program can constrain the paraphrases to matching syntactic structures. In selecting the output paraphrases, the CB program uses a language model to score the fluency of the paraphrase and its immediately surrounding words.

CB Program Adaptations: Several adaptations were made to the CB program to create useful paraphrases for the SCREAM Lab MT system, including the introduction of multiple paraphrases in a single sentence, and restrictions on elements such as digits that should not be changed.

Allow Multiple Paraphrases Per Sentence: The CB program was modified to allow multiple paraphrases per sentence. The revised program takes the top 10 or 20 paraphrases generated by the syntactically constrained process, and permutes these with 1, 2, or 3 changes per sentence, removing any duplications. In cases where only a few or no paraphrases were created, the program repeated the existing sentence(s). An interface was written to specify the number of paraphrases to allow per sentence, and to report statistics on the language model scores and the number of repetitions used for each sentence.

Measure LM Probability across Entire Sentence: The LM gives statistical information of the likelihood of certain words occurring together. In order to allow multiple paraphrases per

sentence, the program was adapted to measure the LM score across the entire sentence, instead of just the revised section.

An error was corrected in the paraphrasing software, enabling the program to read the order of the model directly from the LM file.

Prevent Paraphrasing of Numbers, Dates, Punctuation, Function Words: An examination of the paraphrase output showed that numbers were being changed. The program was modified to prevent the paraphrasing of digits, numeral words, dates, and punctuation, which should be retained to preserve meaning. Other restrictions were added to prevent the paraphrasing of a word to punctuation [and>!], or the paraphrasing of a content word to a function word (article, conjunction, or preposition). This last restriction is necessary when using the LM to weight paraphrase choices, since the function words occur with higher frequency and are therefore preferred over more meaningful paraphrases. For example, without restrictions, the system paraphrased *of thinking to of*. With restrictions, the system paraphrased *of thinking to of thought*, a more useful variation.

Parsing: The Bikel parser [9] was applied to English input to derive syntactically constrained paraphrases. In the absence of parsing capability for other languages, the paraphrase interface was set to allow the creation of non-constrained baseline paraphrases.

Application: The CB paraphrasing program was used to create paraphrases of the English references for the WMT data. Paraphrases were tested in various configurations, including 1, 2, or 3 paraphrases per sentence, with various LMs, and with or without syntactic constraints. Syntactically-constrained paraphrases with multiple paraphrases per sentence were found to be the most useful. Later work also applied the paraphraser to the IWSLT French data as reported in Task Order 13 [10].

Evaluation: Paraphrasing did not improve results as expected; while changing the configuration could improve the scores, paraphrased systems never exceeded the baseline scores. Some problems that kept paraphrasing from succeeding include a limited amount of variation and the creation of antonymic paraphrases.

Amount of Variation: The amount of variation created by the paraphrasing process was analyzed using n-gram matching against the original sentences with the BLEU metric. The amount of variation among four human translators was also examined, and the references provided by one human translator were automatically paraphrased, and then scored against the references created by the three other translators.

In general, automatic paraphrasing did not provide enough variance from the original sentence. Some changes were minor (such as *a > the*), and even allowing multiple changes per sentence did not achieve much variety.

Problem with Antonyms: Examination of the paraphrase output reveals that the pivot system permits the paraphrasing of antonyms, such as *can > can not* or *agree > disagree*. In order to determine the source of the antonyms, a program was written to pull out specified word pairs from the paraphrase tables, and the underlying word alignments were examined. Antonymic paraphrasing of content words can occur because antonyms are usually similar in part of speech and basic semantics (e.g., *north* and *south* both specify locations), and thus occur in similar environments. Antonymic paraphrases involving negation can occur because negation may be expressed in another part of the sentence. For example, the following phrases express similar

meanings, giving the potential for the pivot process to extract the underlined words as “paraphrases:”

if you can not
unless you can
it should be remembered
it should not be forgotten

Because of these contextual dependencies, additional work will be needed to detect and prevent antonymic paraphrasing.

Normalization: Paraphrase creation can also be influenced by tokenization and by the use of British or American English spelling. A mismatch between the input data and the Europarl data used to create the paraphrases can limit the creation of paraphrases, since the words in the paraphrase table may not match the input.

Paraphrasing Using Joshua: Joshua is a hierarchical phrase-based translation system, within which the subsystem Thrax operates to extract grammar rules. Thrax is usually used to extract a translation grammar (e.g., English>French), but can also be set to extract paraphrases using a pivot process similar to that in the CB paraphrase system (see Section 2.1.17). Syntactically-constrained paraphrases can be created by applying a parser for the target language.

Joshua Program Adaptations: Since the Joshua program is still under development, there were a few errors that required correction. Other adaptations provided the optimization and scoring formats required for the SCREAM Lab experiments.

Tokenization Differences between Joshua and Moses: Joshua accomplishes tokenization with a spelling normalizer, followed by application of the Penn treebank tokenizer. An error was corrected in the operation of the Joshua normalizer (see Section 2.1.17).

The Joshua pipeline was also modified to allow specification of either the Penn treebank tokenizer or the Moses tokenizer. An identity function was written to stand in for the spelling normalizer when using the Moses tokenizer option. A Russian recaser was trained and integrated in the Joshua pipeline.

Encoding Problems in Normalizer Script: The Joshua normalizer was apparently written on a Windows system and corrupted in the Linux version, with certain Unicode characters being converted to strange sequences. Those lines were re-written using the Unicode codepoints for the specified characters. The corrupted script produced an unusual side effect for rules involving the non-breaking space 00A0. The script fails to identify 00A0, and also has the side effect of converting the French character à to the encoding sequence \xc3. These \xc3 elements persist in the output of the translation. This happens because the French character à is Unicode \u00E0, hexadecimal \xc3\xa0. The normalizer should look for the nonbreaking space \u00A0 but in its corrupted form it looks for \xa0. Coming to à, it removes the \xa0, leaving \xc3 behind. The revised version of the normalizer using the Unicode codepoint \u00A0 functions as intended.

A localization problem was also identified, in which directional apostrophes were supposed to be normalized to single apostrophes between the alphabetic characters [a-z], and converted to quotation marks elsewhere. This fails for languages with accented characters or other characters outside the [a-z] range. The perl character class \p{Letter} was used to extend the application of the normalizer.

old rule: s/([a-z])'([a-z])/\$1\'\$2/gi;
new rule: s/(\p{Letter})'(\p{Letter})/\$1\'\$2/gi;

<u>Input</u>	<u>Old Output</u>	<u>New Output</u>
l'on	l'on	l'on
l'éducation	l"education	l'éducation

Corrected Problems with Glue Grammar: In the 4.0 release of Joshua, the “glue” rules that allow access to the grammar are not functioning correctly. As a work-around the pipeline script was modified to use an older version of the glue rules instead.

At the suggestion of the developers, work was shifted to the development version of the decoder, which corrects the problem with the glue grammar. The development pipeline script required revision to correct some errors in variable naming and parameter setting, and to copy a missing configuration file when decoding. Scripts were written to call the pipeline in various configurations.

Multiple Optimization Runs and Multi-Bleu Scoring: The pipeline was revised to enable multiple optimization runs, and code was added to calculate the mean and standard deviation for the BLEU scores. Code was added to calculate multi-bleu scoring, in addition to the NIST BLEU scoring originally provided.

Hiero and SAMT Grammars: Experiments were run with the Joshua Hiero grammar option, with Joshua tokenization and then again with Moses tokenization. Another set of experiments were run using the Joshua SAMT grammar option. The initial SAMT scores were lower than the Hiero scores, so further work was conducted using Hiero only.

Span Limit /Distortion: Joshua contains a span limit that affects the number of words that can be considered in the creation of a grammar rule; this parallels the distortion limit in the Moses system. In initial testing, changes in the span limit were found to cause very minor changes in the scores of the Russian-to-English MT, with higher span limits correlating with higher scores.

Change to Joshua 5: Work shifted to the new version of the program, “Joshua 5.” Adjustments were made to the Joshua 5 pipeline to reflect previous adaptations. Some minor problems were identified and corrected in the supporting programs. A baseline run with Russian-to-English data showed a slight improvement over the “Joshua 4” system.

Application: The Joshua system was first tested with a translation task: Models were trained to translate from English to French; those models were then used to decode new English input. The Joshua decoder was then applied to create French paraphrases tables, by “translating” French to French with a paraphrase grammar.

An interface was written to take the output of the Thrax paraphrase tables and create paraphrased sentences. A program was written, “*thrax2tabExp.pl*,” to convert the output of the Thrax module to the tab-separated format needed for lattice generation. This conversion removes the syntactic annotation and converts negative log probabilities to standard [0,1] probabilities.

For example, the Thrax output:

[A] ||| iconique ||| typique ||| p(f|e)=5.86363

becomes:

iconique typique 0.0028409...

Adjustments were made to the language modeling process to prevent underflowing values when using the Joshua-generated paraphrase grammar as input to the existing paraphrase generation program. Information from the language model was called using the new utility “*query_interp*.” As part of other language model research, (see Section 2.1.6.) a program based on code from the KenLM toolkit was adapted to query the probability of a specific ngram from a language model. This program is conveniently a direct replacement for the 32 bit modified SRILM ngram binary distributed with the paraphrasing software.

Evaluation: The paraphrases created via Joshua seem limited: The resulting N-Best list seems to lock in on certain paraphrases. For example, the phrase *par contre* “on the contrary,” was translated as *mais* “but” in all 300 translations of the sentence, despite the presence of 158 possible paraphrases for *par contre* in the paraphrase grammar, such as *alors que* “while,” *tandis que* “while,” and *au contraire* “on the contrary.”

Paraphrasing Using the PPDB: The PPDB is a database of 637,000 English-to-English paraphrases extracted from bilingual parallel corpora according to the pivot method (see Section 2.1.17). The parallel corpora include Europarl as well as several other sources, including the News Commentary, UN, and JRC Acquis data. The paraphrases in the database are listed with LM probabilities and paraphrase probability values.

Adaptation: In order to apply the PPDB information, a program was needed to adapt the format of the database and create the actual paraphrases.

Creating Paraphrased Sentences: A program, “*createParaTableFromPPDB.pl*,” was written to collect information from the PPDB in order to create a paraphrase table in the format used with the CB paraphrase system (Section 2.1.17). The new paraphrase table lists the original phrase, the new phrase, and the paraphrase probability, $p(f|e)$. It was also necessary to remove control characters from this data. A script was written, *use-PPDB-paraphrases-2013.sh*, to call the existing paraphrasing program, “*KBaselineParaphrases2013.java*,” with the paraphrase table and a LM built from the TED training data.

Information from the LM was called using the new utility *query_interp* (See Section 2.1.17).

Escape XML Mark-Up: The LM had to be reformatted using a version of the Moses character escaping program, “*escape-special-chars-no-removals-no-angle-bracket-conversion.perl*.” Special characters are converted using the Moses escaping conventions; in de-escaping, tab characters and angle brackets characters are protected to preserve the tab-formatting and sentence notation *<s>*.

Paraphrasing the dev files required pre-processing of files to remove XML mark-up, tokenize, and lowercase; and then post-processing to truecase, detokenize, and restore XML mark-up, using the original file as a template. Existing programs were applied for pre-processing and post-processing, with post-processing adjustments to the XML mark-up using “*removeTagsFromDisambig.perl*” and “*moses2mteval-ref-step2.perl*.”

Application: Paraphrased Dev References for Russian, Chinese, Arabic, and Farsi:

Paraphrases were created for the English dev files in the Russian-to-English, Chinese-to-English, Arabic-to-English, and Farsi-to-English MT experiments. By paraphrasing each dev sentence 3 times, a total of 4 references were derived against which the MT output could be scored.

Evaluation: Using paraphrases to expand the dev references yielded higher scores during optimization, when the dev output is compared against the multiple references, but did not improve scores overall.

These results were compared with experiments using paraphrases created from the europarl LM. Some different word choices were made (e.g., *every* > *each* vs. *every* > *all* with europarl).

2.1.18. Reverse Palladius (RevP) Project

The RevP project was created under Task Order 14 [7]. This project provides a user interface for applying automatic correction of the translated pinyin spelling of Chinese names in Russian text. The correction is needed to reverse the “Palladius” mapping with which Russian speakers write Chinese sounds. The program was previously presented at the 2012 conference of the Association for Machine Translation in the Americas (AMTA).

In the current effort, the RevP program was reviewed and transitioned to the user. Consultation with the users provided the opportunity to review the final program changes and collect suggestions for future development, such as using Chinese syllabic constraints to distinguish Chinese names from Russian OOV words. During implementation, questions from the user were fielded about the Systran interface, file formats, and the RevP Protect English function.

A comparison was made with Google Translate, which has improved its treatment of Chinese names in Russian text. The Google translate program now handles both basic and inflected forms of the cosmonaut names used in the RevP test data, but it fails to correctly translate less common names, suggesting that the improved performance is based on additions to its named entity lists, as opposed to a sound mapping.

2.1.19. Pashto-French-English Translation Project

Work continued on a project to produce translations of Pashto audio data into French and English. The initial Pashto>French translation data was received, and a test file was translated into English. The English translator identified certain problems involving idiomatic phrases and possible variation in the transliteration of names. A review of the audio files also detected problems with the transcription of overlapping speech.

Need to Standardize Transliteration of Names: It is necessary to be aware of differences in transliteration conventions in English and French. For example, the name of Iran's ambassador to the International Atomic Energy Agency has these variants:

Pashto	علي اصغر سلطاني /āly aṣṣyr slṭany/
French	Asghar Sultani
English	Ali Asghar Soltanieh

It was decided for now to have the French>English translators keep the names in the same form as the French document. The most desirable solution would be to have the Pashto translators tag names as they transcribe the audio; this would enable the compilation of a table of the names in Pashto script, French transliteration, and English transliteration.

For one news item, the English>French translator identified an existing English version on the Voice of America (VOA) English website. This provided an interesting comparison between the original English and the more literal translation from French into English. No other English-language postings were found for this dataset.

Examination of the English VOA news item identified a mis-translation of an American name: *Dr. Paula Holmes-Eber* was written as *Docteur Paula Plain Hubbard*, and later in the file *Holmes-Eber* was written as *Helen Hubbard*. The French and Pashto texts were reviewed, as well as the Pashto audio for that section, and it was determined that there was an error in transcribing the Pashto audio. This suggests that non-Pashto names may cause problems for the transcribers; tagging the names would support error detection and correction.

Need to Indicate Overlapping Speech in the Transcriptions: A review of the audio files indicated the presence of overlapping sections involving either speech or environmental sounds. Typically, when the broadcast includes audio from a non-Pashto speaker, the foreign speaker begins his or her remarks, and then this is overlaid with the voice of the interpreter. This voice-over is not indicated in the metadata; the beginning section is marked as foreign speech, but the voice-over section is marked as Pashto. These sections will need to be excluded when building LMs.

2.1.20. Review of Previous Work and Available Tools for Chinese MT

A summary was prepared reviewing previous efforts in Chinese MT. These efforts included word re-ordering to anticipate English word order, the removal of articles from the English text to better parallel the Chinese text, comparison of Chinese word segmentation techniques, and comparison of Chinese parsers.

Instructions and examples were prepared for the use of Chinese word segmenters, NE taggers, parsers, and word alignment editing software. Specific software included the Janya NE tagger for Chinese, the General Architecture for Text Engineering (GATE) development environment and the Chinese plugin for word segmentation and named entity detection. Literature was reviewed on the GATE/Java Annotation Patterns Engine (JAPE) interface and on transliteration of Chinese named entities. Available Linguistic Data Consortium (LDC) NE lists were identified and reviewed for encoding problems.

Objectives were outlined for lattice programs to generate word segmentation variants, and for setting up Chinese-to-English translations within automatically generated named entity tags.

2.1.21. SRILM

The Maximum-Entropy, Random-Forest, and SCREAM patches for SRILM were extracted into separate branches under version control to allow for faster integration of updates from their

official sources. The Maximum-Entropy patch spends the majority of its time inside the Limited-Memory Broyden-Fletcher-Goldfarb-Shanno (LBFGS) library, so the library was rewritten to take advantage of the Graphics Processing Unit (GPU). To overcome the additional time imposed by transferring data to the GPU, the size of the input data needs to be above 38,000 before the GPU becomes a faster alternative to the Central Processing Unit (CPU). While the GPU version of LBFGS is around 30% faster, GPU cards do not currently have enough on-board memory available to support the scale of data that we use in the SCREAM Lab.

Additionally, support was added to SRILM for generating and evaluating skip models with arbitrarily sized skip-gaps. This feature was added very quickly and retains backward compatibility with previously created skip models, but suffers from a few drawbacks. Most notably, the back-off weights no longer have meaning because the skip-gap eliminates some of the words in the context. To generate accurate back-off weights would require a standalone project outside the confines of SRILM.

2.1.22. Trigger-Based Lexicon Model

A modified version of the Trigger-Based Lexicon Model described in “Extending Statistical Machine Translation with Discriminative and Trigger-Based Lexicon Models” [11] was created for the SCREAM environment. While the original paper described a system for processing bilingual data (a source and a target document), we allow for the model to be applied to a single input document. An option was added to restrict the triggering behavior to work in a left-to-right manner (i.e. a group of words at the end of the sentence could not trigger a word at the beginning of the sentence). That option leaves the first few words in a sentence without a probability, so support was added for a standard N-gram model to be used in situations where the trigger model does not apply. We also eliminated the restriction on the context size (the number of words that ‘trigger’ another word).

The consequence of eliminating the restrictions imposed by the paper was that the number of word combinations increased, along with processing time and storage requirements. A typical trigger model is around 90 GB, uncompressed, on disk and needs approximately 200 GB of memory at runtime. The enormous amount of time needed for training required us to create a distributed evaluation process. The distributed process creates a series of partial models, from which a complete model can be aggregated without any loss of precision.

The amount of time required for loading trigger models also compelled us to look into alternative approaches for storing the model. An SQLite-backed model was extremely quick to load, but evaluation was much too slow due to the latency on each query. A version of the training code was written in lisp (elisp), and while the code itself was 90% shorter, the execution time increased 400%. Finally, the model loading code was rewritten to be multithreaded and pipelined, which reduced load times by approximately 80%.

2.1.23. Recurrent Neural Network LM (RNNLM)

A defect in how word classes were being internally identified caused the RNNLM to prematurely truncate its formulas. The model assumed that ‘0’ was an invalid class identifier, when it is not. This caused any word in that class to use a slightly different calculation than the rest. The result is that the activation value from class ‘0’ would be mildly suppressed.

2.1.24. Neural Probabilistic LM (NPLM)

The NPLM project was distributed with a build system that was inflexible and prone to error. These shortcomings were corrected and additional options were added to create valid statically linked binaries. In addition, the evaluation stage of the model was modified to accept N-Best lists as valid input. This allowed NPLM to be more easily used in a pipeline.

2.1.25. Translation Web Services

The SCREAM Lab has numerous translation packages available, and the desire was to incorporate methods for these packages to communicate with each other or to communicate back to a common web-based application. Most of these packages allow for Simple Object Access Protocol (SOAP) / Web Services Distribution Language (WSDL) interfaces which were advantageous in creating new tools for the Lab.

SOAP and CyberTrans: CyberTrans is a MT System produced by the DoD's Center for Applied Machine Translation (CAMT). It incorporates the commercial Systran translation system as well as Gister and Motrans, two US Government-produced systems, allowing for translation of 78 different languages. CyberTrans provides tools for automating translation, language identification and pre/post processing of text with built-in web services.

While a powerful translation tool, the front end of the CyberTrans interface locks a user into manually checking its many options and mandating a classification system that is embedded into the translated text. A portion of the CyberTrans interface is shown in Figure 1.

SOAP and PHP: A PHP/SOAP solution was investigated due to all the various MT solutions



Figure 1: A Portion of the CyberTrans Interface for Translation

and a need to manipulate the translated results. A previous community-built SOAP extension for PHP had become outdated. New hooks into the pre-existing functionality were developed for enabling the latest SOAP web services.

Initial SOAP testing for translation began with single lines of text sent to the Systran MT system and the result printed out to the screen. The preliminary results were promising and development continued by creating a rudimentary interface for pasting in text and selecting from 4 basic languages for translation: Arabic, French, German and Spanish.

The PHP scripting evolved to allow uploading of a text file into an array structure. Each line of text is translated individually. By the end of the process the array contained the complete, translated document.

SCREAM Lab Translation Web Services: A tool for simplifying the translation process with large amounts of text was developed so that a user could bypass the enforced classification tagging, and issues dealing with large files would be avoided.

CyberTrans Application Programming Interface (API): The CyberTrans API allows for functionality through web services. There was an issue with CyberTrans throwing a fatal error when large files of text exceeding 200,000 lines of text were uploaded for translation. CyberTrans also stamps a classification level label into its translation results, causing undesired labels between the foreign language and translated text.

The CyberTrans Translation Service web service hooks into virtually every option available through the interface, allowing for automated manipulation of those parameters.

Architecture: Based on experiments communicating to CyberTrans via SOAP/WSDL, a more robust means of communication was desired for translating large text files.

The basic idea was to create a front end for uploading large text files, selecting the languages for translation (or select the HOTSPOT language identifier), and selecting which dictionary to use. The file is uploaded to the web server and broken into manageable 20,000 line chunks of text and then submitted to CyberTrans for translation. Once each chunk of text is translated and returned, the server stores the result and sends the next chunk of text. When the entire file is translated, the server removes labels inserted by CyberTrans, combines the translated chunks and returns a complete, translated file.


As a quick error check, the line counts of the original and translated file are compared. If the line counts are different, the user is notified of a potential error. Finally, the server creates a link in the interface to the translated file.

Front End Graphical User Interface (GUI): Through the CyberTrans Translate Service getParameters function, a list was available translation engines, languages, and dictionary options was generated. This information was used to create drop-down selection options in the interface as shown in Figure 2.

Moses WSDLeXensible Markup Language (/XML)-Remote Procedure Call (RPC): The Moses MT system can be run as a server and service clients with XML-RPC libraries. To extend the dynamic translation capabilities within the SCREAM Lab, various scripts and processes were created to take advantage of the Moses MT Server.

Moses Server XML-RPC: By invoking the XML-RPC, protocol distributed clients can use the Moses Server decoder service. By default Moses Server listens on Port 8080 and the sample client that is included is written in Perl.

SCREAM Lab Translation Web Services



Language (if known): French

Dictionary: general-Gister

Text Dump Translation

anslate
Submit

general-Gister
 general-ascii-Gister
 general-Systran5
 automotive-Systran5
 aviation_space-Systran5
 chemistry-Systran5
 colloquial-Systran5
 computers_data_processing-Systran5
 earth_sciences-Systran5
 economics_business-Systran5
 electronics-Systran5
 food_science-Systran5
 life_sciences-Systran5
 mathematics-Systran5
mechanical_engineering-Systran5
 medicine-Systran5
 metallurgy-Systran5
 military_science-Systran5
 naval_maritime-Systran5
 photography_optics-Systran5

Text File Translation

anslate
Browse...

anslate
Click to

Figure 2: Example of Drop-Down Selection of Language, Dictionary and Translation Engine

Initial single line testing of the default Moses Server setting was promising and development started by extending the architecture to handle the translation models being used in the SCREAM Laboratory and using PHP to extend its boundaries.

After some investigation, the most promising open source library for XML-RPC to run on PHP was the Incutio XML-RPC Library,⁸ that incorporated both client and server classes with automatic type conversion.

Moses Server WSDL with EuroParl: Using the thoroughly tested EuroParl [12] French-to-English and English-to-French LMs, single line phrases were translated via command line. After successful command line translation of single line phrases, a front end was developed to send phrases to the Moses Server with a choice of language to use for translation. The interface for the single line phrase translation is shown in **Figure 3**. This was all based on underlying PHP script as the delivery system.

To send full text documents across the network for translation with Moses Server, a WSDL/SOAP solution was devised. A client WSDL file was defined to send the parameters of the file to be translated as well as the server, port, and extraneous services that might be used for tokenization and casing of the text. The client calls the SOAP service that initiates the binding of the values to the parameters and calls on the PHP script that will process the translation.

⁸ <http://scripts.incutio.com/xmlrpc>

A PHP script was written from the original Moses Server XML-RPC service that splits the text file into an array of strings and sends them one by one to the remote procedural call. As Moses Server returns each translated string, PHP stitches the strings back into file form and runs a line count check to make sure that the sentence alignment is correct.

(all input is re-formatted to lower case)

Language: French > English ▼

Phrase: Tout le monde a été très enthousiasmé de cette découverte archéologique

Translation: everyone was very enthusiastic about this archaeological discovery

Submit Reset

Figure 3: Basic, Single-Line Phrase Interface Using XML-RPC with Moses Server

Once the script confirms that translation is complete and sentence alignment is correct it will save the file and send a call to the translation SOAP service, which then sends a message back to the user interface with a link to the translated file. The interface script then opens the translated file and shows the translated text next to the source text.

Perl/SOAP Speech-to-Speech (S2) Translation: With the successful pairing of WSDL and SOAP as a translation system, there was interest in adapting this process to work with the SCREAM Lab's S2S application. S2S is an application for capturing spoken phrases, converting to text via ASR, translating the text, and converting the text back to speech.

A Perl script was developed to run as a bridge for the translation. The SOAP::Lite Perl module was installed to enable the protocol to speak to CyberTrans' web services. The script was successful in speeding up the translation process.

2.1.26. Evaluation and Scoring

MT must constantly be evaluated and scored in order to recognize areas for improvement. These methods create staggering amounts of data and various methods are used in keeping the input manageable and meaningful.

MT Eval System: In 2005, CMU developed MT Eval, a web application for users to define their own test sets and experiments from MT and have their scores automatically calculated from a translation hypothesis.

Using the MT Eval application to calculate scores, new functionality to analyze and view experiment data was developed in the Lab.

Architectural Updates: Over time, some scoring calculations were updated resulting in CMU releasing updated Java Archive (JAR) files for the MT Eval application. This required updates to the SCREAM Lab MT Eval web-based user interface and testing of the new functions included in the JAR file.

The MT Eval server was also updated to allow a means to view the translation hypothesis with its unique scoring on a line-by-line basis. PHP script was developed to take this output and generate a Portable Document File (PDF) file that could be viewed online or downloaded for offline study.

Front End Updates: A user defines a new test set by choosing the reference text, source file and target/source languages. Experiments are created by choosing the text set, preprocessing steps (lower case, punctuation, etc.) and which scores to calculate. Once the test set and experiment are created a user can then submit a translation hypothesis and have the scores calculated.

A PHP script was developed to color code n-gram text and incorporated into the hypothesis viewer. Each unique color represented a unigram, bigram or trigram and could be turned off and on dynamically.

Experiment Reader: The Experiment Reader web application was developed to help sort through the enormous amount of scoring data created during MT experiments. There needed to be way to view and sort the scores, statistics, dates and configuration files generated with each translation process.

Architecture: The scoring data is saved within stats files during the MT process. Scripts were created to read through selected directories and parse out the scoring data. This information was then saved out to a MySQL database so that the scores could be individually displayed and sorted on the front end.

A process was developed for re-scanning the directories and looking for updated or new information and mirroring those changes within the database.

Front End GUI: A basic front end was initially developed to control how the data could be displayed and sorted. Each configuration and stats file displayed is also a link to the actual file so that the user can bring up the complete file for viewing.

As the interface was used and the data sets grew larger, new enhancements were requested for sorting and displaying the information. JQuery⁹ was adopted to help in the management of the front end. JQuery is a free, open source JavaScript library for dynamic update and control of web pages incorporating various features of client-side scripting.

Each category (directory, file, configuration file and numerous scores) can be selected for sorting. Directories and various score categories can be selected through drop-down selectors to allow the user full control of the scores they would want to review. At any time a user can select a directory to be rescanned for new scoring data and export it all to a comma-separated file for viewing offline or to integrate into other data manipulation software.

It was also necessary that other peripheral information be available and easily accessible. If a user was browsing the scores and needed more specific information on which configuration file was used they can select the configuration file and view its contents.

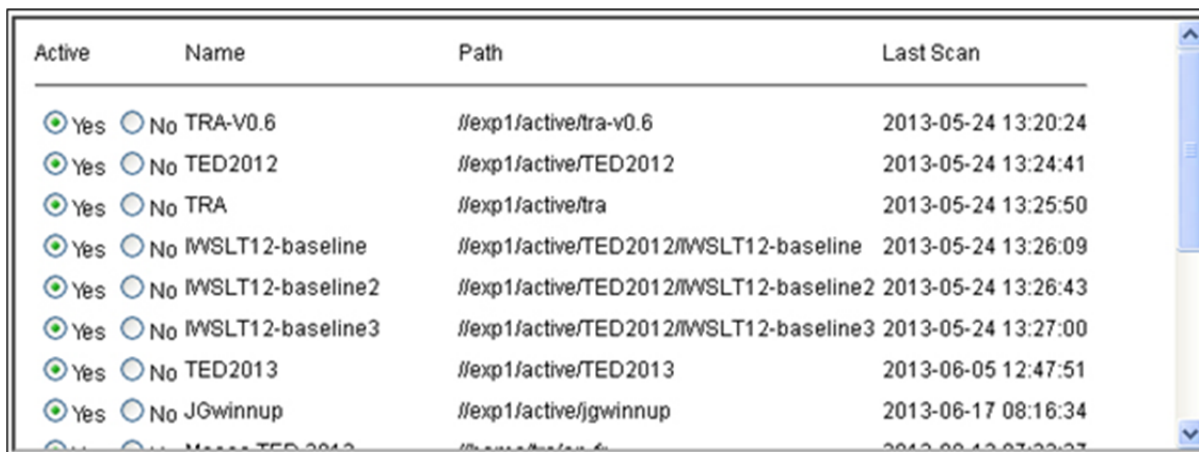
Search: An often requested update to the front end was the ability to search and filter information so that the display was easier to manage and read. A search function was added to enable filtered results from the overall score sets or by a specific column such as the file name or configuration file.

Additional search functionality was added to allow a secondary sort parameter. This allows sorting on an initial value such as BLEU Score and then sort by the Meteor Score and see if similar configurations rise to the top.

⁹ <http://jquery.com/>

Administrative Tools: The next phase in the development of the Experiment Reader application was a means for self-regulation of the information.

Tools were developed for creating new directories, archive directories no longer needed, create comma separated value files of the results, and to scan any and all directories for changes. The tool for archiving directories is shown in **Figure 4**.



Active	Name	Path	Last Scan
<input checked="" type="radio"/> Yes <input type="radio"/> No	TRA-V0.6	//exp1/active/tra-v0.6	2013-05-24 13:20:24
<input checked="" type="radio"/> Yes <input type="radio"/> No	TED2012	//exp1/active/TED2012	2013-05-24 13:24:41
<input checked="" type="radio"/> Yes <input type="radio"/> No	TRA	//exp1/active/tra	2013-05-24 13:25:50
<input checked="" type="radio"/> Yes <input type="radio"/> No	IWSLT12-baseline	//exp1/active/TED2012/IWSLT12-baseline	2013-05-24 13:26:09
<input checked="" type="radio"/> Yes <input type="radio"/> No	IWSLT12-baseline2	//exp1/active/TED2012/IWSLT12-baseline2	2013-05-24 13:26:43
<input checked="" type="radio"/> Yes <input type="radio"/> No	IWSLT12-baseline3	//exp1/active/TED2012/IWSLT12-baseline3	2013-05-24 13:27:00
<input checked="" type="radio"/> Yes <input type="radio"/> No	TED2013	//exp1/active/TED2013	2013-06-05 12:47:51
<input checked="" type="radio"/> Yes <input type="radio"/> No	JGwinnup	//exp1/active/jgwinup	2013-06-17 08:16:34
<input checked="" type="radio"/> Yes <input type="radio"/> No	Mass-TED-2013	//exp1/active/Mass-TED-2013	2013-06-17 08:22:27

Figure 4: Directory Archival Tool

iBLEU Integration: iBLEU is a JavaScript based tool created by Nitin Madnani¹⁰ to examine the output from statistical MT and give it a BLEU score down to the segment level, as shown in Figure 5. It can also submit the segment to Google's and Bing's translation services to compare the output.

With the SCREAM Lab's standalone network, the iBLEU code was updated to allow for translation requests to be sent to a Systran server for translation comparison.

¹⁰ <http://desilinguist.org>

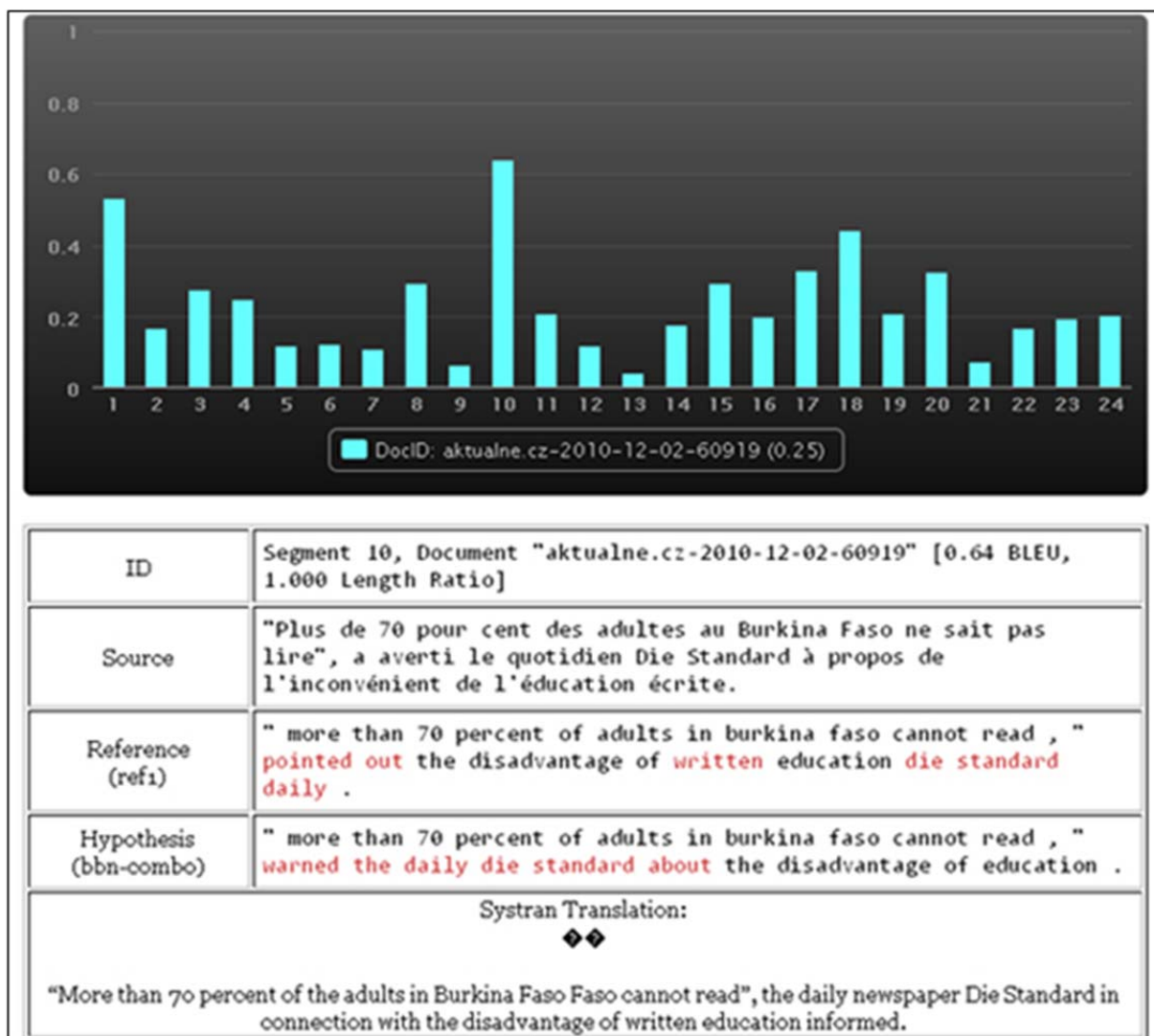


Figure 5: Interface from IBLEU Scoring a Specific Segment of Text

2.1.27. Summary Figure

The MT efforts reported here focus on the reduction of variation in the input. In general, Phrase-Based MT has trouble processing infrequent variant forms. Variations that occur with high frequency can generally be translated, since there is sufficient training data for each variant, but infrequent variants may remain untranslated. For example, if a training set with English as one of the languages contains frequent instances of both the word *color* (American spelling) and the word *colour* (British spelling), then both can be translated into the other language. On the other hand, if there are only one or two instances of the British spelling, then spelling normalization of *colour* to *color* might be needed to translate the British form.

A wide range of techniques was applied to reduce variation. Generally, the following approach was found to be valuable: (1) measuring variation, and restricting corrective techniques to infrequent variants; (2) applying frequency-based techniques to select useful forms; and (3) using lattices to represent and apply information from different versions of the data.

Measuring variation is important, because normalization reduces information, and should only be applied when the variations cannot be handled by the MT system. Frequency-based techniques help the system by targeting normalization efforts to unknown or infrequent variants, and by ensuring that the resulting normalized forms are found in the data. Lattices retain information that is lost when using a single-best variant, while requiring less storage space than listing all the variants independently.

Performance improvements, functionality enhancements, and error corrections were performed on various software packages which are actively being used in the SCREAM Laboratory. In addition, a custom trigger-based lexicon model was created based on a 2009 paper [11].

Various methods were incorporated to allow the Lab's translation services to communicate with each other. Scripts were developed to take advantage of the CyberTrans API and perform simple SOAP/PHP single-line translations. This technology was then adopted into a web-based application to translate documents containing hundreds of thousands of lines of text.

A system was developed that enabled Moses Server to translate text across the network using the XML-RPC protocol. Once this process was proven successful it was also developed into a web-based translation service for larger documents.

A Perl script was developed to act as a bridge for the S2S system and the CyberTrans SOAP-based translation process.

The MT Eval system was modified to allow for viewing of the translation hypothesis in a more defined manner and to allow for n-gram color coding of the text. Updates were also made to various scoring files as they were updated by the MT community.

An Experiment Reader was developed to manage the scores, statistics, dates and configuration files generated with each translation process. Functionality was developed for self-regulation of the files, filtering and searching of data, and the integration of the iBLEU visualization tool.

2.1.28. Recommendations for Future Work

It would be beneficial to develop a pipeline so that submitting a document and selecting its language of origin would send it to all possible translation services available within the Lab and view the results side-by-side.

It might be beneficial to continue development of the Experiment Reader to allow for more fine-tuned viewing of the scoring and administrative functions. Currently iBLEU has to be directed to the hypothesis and reference files and manually sent to the scoring process. It would be an improvement to enable dynamic formatting of the hypothesis and reference files that are automatically created and sent for immediate results.

General-Purpose Graphics Processing Unit (GPGPU) cards with higher memory capacity may allow data at our scales to be processed, especially with respect to neural networks. In addition, accelerated processing units from Advanced Micro Devices (AMD) that have shared memory space between the CPU and the GPU may provide a significant boost in performance without data transfer overhead.

2.2 ASR

This section discusses the tools and models that were developed for ASR. Section 2.2.1 describes a script that was created for converting acoustic models to Sphinx-4 format. Sections

2.2.2 – 2.2.4 describe the Dari and Pashto, Mandarin, and English Speech Recognition Systems (SRS) that were developed. Section 2.2.5 presents a summary of the results obtained, and Section 2.2.6 provides recommendations for future work.

2.2.1. Sphinx-4 Models

A Perl script was developed for converting Hidden Markov Model (HMM) ToolKit (HTK) [13] acoustic models to Sphinx-4 American Standard Code for Information Interchange (ASCII) format [14]. Support is provided for state clustered across word triphones, provided that all states are modeled using the same number of mixture components. Feature transforms are not currently supported.

This program was validated by converting a previously developed English SRS from HTK to Sphinx-4 format. The HMMs were trained on 18 hours of speech produced by 80 speakers from the Wall Street Journal (WSJ) corpus [15, 16]. Phonemes were modeled using state clustered across word triphones, and the final HMM set included 3000 shared states with 16 mixtures per state. The models were discriminatively trained using the Minimum Phone Error (MPE) criterion. The feature set consisted of 12 Mel-Frequency Cepstral Coefficients (MFCCs) plus energy with mean normalization applied on a per utterances basis. Delta, acceleration, and third differential coefficients were appended to form a 52 dimensional vector, and Heteroscedastic Linear Discriminant Analysis (HLDA) was applied to reduce the feature dimension to 39. The SRILM Toolkit [17] was used to estimate a trigram (LM on the WSJ training transcripts and the 1987–1989 WSJ Continuous Speech Recognition (CSR)-LM1 corpus. This system was evaluated on the Hub 64k task from the November 1993 Advanced Research Project Agency (ARPA) CSR test corpus [16] using Sphinx-4, the HTK large vocabulary continuous speech recognizer HDecode, and the Julius speech recognition engine.¹¹ The following Word Error Rates (WERs) were obtained: 15.8% with Sphinx-4, 15.3% with HDecode, and 16.7% with Julius. Sphinx-4 yielded a 0.3% increase in WER compared to HDecode, and a 1.4% decrease in WER compared to Julius.

2.2.2. Spoken Language Communication and Translation System for Tactical Use (TRANSTAC) Dari and Pashto

Dari and Pashto SRS were developed on the TRANSTAC corpus. This corpus includes 110 hours of Dari spoken by 192 speakers and 130 hours of Pashto spoken by 156 speakers. There are two collections of recordings for each language: the first consists of conversations between an English interviewer, an interpreter, and a Dari or Pashto respondent; and the second consists of conversations between a Dari or Pashto interviewer and respondent. Each conversation is provided as a single audio file with an utterance level transcript.

The first step for developing an SRS on this corpus was to segment the audio. Two methods were investigated for segmenting each conversation: (1) using the time marks as provided, and (2) merging sequential utterances from the same speaker. Next, HMM SRS were developed on each data set using HTK. Phonemes were modeled using state clustered across word triphones and each HMM set included 5500 shared states with an average of 12 mixtures per state. The models were trained using Maximum Likelihood Estimation (MLE). The feature set was the same as described in Section 2.2.1, except that HLDA was not applied. Trigram LMs were

¹¹ Available at: http://julius.sourceforge.jp/en_index.php

Table 1: WERs Obtained on TRANSTAC Dari and Pashto

Merged Utterances	Dari	Pashto
No	15.8	14.5

estimated on each data set using the SRILM ToolKit. In order to identify possible transcription errors, each system was evaluated on the entire data set using HDecode and the recognition hypotheses were compared to the transcripts. Table 1 shows the WERs obtained on each language. Merging sequential utterances from the same speaker reduced the WER of the Dari and Pashto systems by 0.5% and 1.0% respectively.

2.2.3. CALLHOME Mandarin

SRS were developed on the CALLHOME Mandarin corpus [18]. The training partition includes 12 hours of speech from 80 telephone conversations; the development and test partitions each include 3 hours of speech from 20 telephone conversations. HMMs were trained using HTK. Phonemes were modeled using state clustered across word triphones, and the final HMM set included 2329 shared states with an average of 12 mixtures per state. The models were discriminatively trained using the MPE criterion. The feature set consisted of 12 MFCCs, plus energy and a pitch feature, with mean normalization performed on per utterance basis. Delta and acceleration coefficients were appended to form a 42 dimensional vector. The pitch feature was calculated using the Entropic Speech Processing System (ESPS) method implemented in the Snack Toolkit.¹² Since continuous HMMs were used, it was necessary to define pitch values over unvoiced segments. This was accomplished using the method described in [19].¹³ A trigram LM was estimated on the training transcripts using the SRILM Toolkit.

¹² Available at: <http://www.speech.htk.se./snack>

¹³ This algorithm was implemented by Mr. Eric Hansen

Table 2: Methods for Applying Discriminative Training to the Callhome Mandarin HMM SRS

Method 1: Baseline System	
1.	Train a single mixture shared state HMM system using MLE.
2.	Increase the average number of mixtures per state by 2.
3.	Re-estimate the models using MLE.
4.	Go to Step 2 if the average number of mixtures per state is less than 12.
5.	Apply discriminative training.
Method 2: Discriminative Training Incorporated Into The Mixture Incrementing Procedure	
1.	Train a single mixture shared state HMM system using MLE.
2.	Increase the average number of mixtures per state by 2.
3.	Re-estimate the models using MLE.
4.	Apply discriminative training.
5.	Go to Step 2 if the average number of mixtures per state is less than 12.

Discriminative training is typically applied as the final step in the HMM training procedure. In this experiment, a second HMM system was trained where discriminative training was incorporated into the mixture incrementing procedure. Table 2 presents a summary of each method. These models were evaluated on the development partition using HDecode. The baseline models yielded a 61.5% Character Error Rate (CER), and the models trained where discriminative training was incorporated into the mixture incrementing procedure yielded a 60.8% CER.

2.2.4. Fisher and WSJ English

This section presents the English SRSs that were developed on the Fisher [20, 21] and WSJ corpora. All systems were trained using HTK and modeled phonemes using state clustered across word triphones. The Fisher HMMs were trained on 1973 hours of acoustic data from 11699 telephone conversations. Models were estimated using 8000, 10000, and 20000 shared states with an average of 28 mixtures per state. The feature set consisted of 12 Perceptual Linear Prediction (PLP) coefficients, plus energy, delta, acceleration, and third differential coefficients. Mean and variance normalization applied on per conversation side basis, and HLDA was applied to reduce the feature dimension to 39. The models were trained using MLE. A trigram LM was estimated on the training transcripts using the SRILM Toolkit and HDecode was evaluated on the speech-to-text task from the 2003 National Institute of Standards and Technology (NIST) rich transcription evaluation [22]. The following WERs were obtained: 38.7% with 8000 shared states, 38.1% with 10000 shared states, and 36.9% with 20000 shared states.

The WSJ SRS was developed using the same procedure as the system described in Section 2.2.1, except that acoustic models were trained on 130 hours of speech produced by 334 speakers and the final HMM set included 4500 shared states with an average of 24 mixtures per state. This system was evaluated on the Hub 64k task from the November 1993 ARPA CSR test corpus using the HDecode and Julius recognizers. HDecode yielded a 10.2% WER and Julius yielded an 11.5% WER.

2.2.5. Summary

This section summarizes the tools and models that were developed for ASR. First, a script was created for converting HTK acoustic models to Sphinx-4 ASCII format. Next, SRSs were trained on the TRANSTAC Dari and Pashto corpus. These models were evaluated on the entire corpus in order to identify possible transcription errors. The best performing systems yielded a 15.3% WER on Dari and a 13.5% WER on Pashto. Third, SRSs were developed on the Callhome Mandarin corpus. Two different methods were investigated for applying discriminative training: Method 1 applied discriminative training as the final step in the HMM training procedure, whereas Method 2 incorporated discriminative training into the mixture incrementing procedure. Method 1 yielded a 61.5% CER and Method 2 yielded a 60.8% CER on the Callhome development partition. Lastly, English SRSs were developed on the Fisher and WSJ corpora. The best performing Fisher system yielded a 36.9% WER on the speech-to-text task from the 2003 NIST rich transcription task. The WSJ system yielded a 10.2% WER with HDecode and an 11.5% WER with Julius on the Hub 64k task from the November 1993 ARPA CSR test corpus.

2.2.6. Recommendations for Future Work

A script was developed for converting HTK models to Sphinx-4 ASCII format. As mentioned in Section 2.2.1, this script does not currently support feature transforms; it would be useful if transforms were supported since they are common in HMM systems. Also, Sphinx-4 does not currently support models with varying number of mixture components per state. This should be modified because HMMs with varying number of mixtures per state can yield an improvement in system performance.

WERs of 15.3% and 13.5% were obtained on the TRANSTAC Dari and Pashto corpus. This may indicate that there are errors in the transcripts. It would be beneficial to correct the transcripts so that more data can be used for training systems.

The best system on Callhome Mandarin yielded a 60.8% CER. Recall that the training partition for Callhome Mandarin only includes 12 hours of speech. The performance of this system may be improved by incorporating additional training data, or using a different modeling approach. Possible suggestions include hybrid Deep Neural Network HMMs [23] or Recurrent Neural Network LMs [24]. The English systems may also benefit from these modeling techniques.

2.3 Speech Synthesis

This section discusses the tools and models that were created for TTS. Section 2.3.1 describes the text-to-speech GUI that was developed for synthesizing speech in real-time, and Section 2.3.2 describes the audio recorder GUI that was created for recording a speech database. Section 2.3.3 presents a tool that was developed for generating phone time alignments of audio files. Sections 2.3.4 – 2.3.7 describe the English, Iraqi, Mandarin, and Spanish TTS voices that were created. Section 2.3.8 presents a summary of the experiments performed, and Section 2.3.9 provides recommendations for future work. All TTS systems discussed in this section were developed using the HMM TTS System (HTS).¹⁴

¹⁴ Available at: <http://hts.sp.nitech.ac.jp>

2.3.1. TTS GUI

A TTS GUI was previously developed in the SCREAM Laboratory using `hts_engine` software and the Tcl/Tk programming language [25]. This GUI allowed the user to select a language and speaker, adjust the `hts_engine` synthesis parameters, modify the pronunciations of words, and synthesize speech in real-time. Support was provided for `hts_engine` models trained on triphone labels, and pronunciations of unknown words were generated using the SONIC spell program [26].

The Tcl/Tk GUI was ported to Perl GTK and several new features were incorporated. First, functionality was added to perform English text analysis using Festival.¹⁵ This enables the use of `hts_engine` models with syllable, accent, stress, part-of-speech, word, and phrase information in the label set. Second, Kevin Lenzo's letter-to-sound program [27] was added as an alternative to `spell`. Lastly, menu items were created to save the generated speech to a file and change the font size of the input text. The main interface, `hts_engine` properties, and lexicon editor are shown in Figure 6.

In addition, pause models were created for each of the 151 `hts_engine` voices used in the GUI. This was done by first sampling 10 seconds of silence from a database and calculating Mel-Cepstrum and Log F0 features. Next, the mean and variance of each feature stream was calculated and added to the set of `hts_engine` probability density functions. Duration parameters were also added that specified mean values of 0.1 to 1.0 seconds. Finally, nodes were added to the `hts_engine` decision trees so that the silence parameters could be used to synthesize pauses with varying durations.

2.3.2. Audio Recorder GUI

An audio recorder GUI was developed using Perl GTK. The GUI prompts the user to record a set of utterances and saves each recording in a separate file. Audio data is recorded using the Microphone class from the Sphinx-4 speech recognizer. Playback of the recorded speech is supported using the PortAudio Real-Time Audio Library.¹⁶ The ØMQ messaging library¹⁷ is used to pass data between the GUI, recorder, and playback programs. Figure 7 shows the main interface of the audio recorder.

2.2.3. Phone Alignment Tool

A Perl script was developed for generating phone time alignments of audio files with sentence level transcripts. Support is provided for any language provided that the following files exist in the program configuration directory: acoustic models in HTK format, a Sequitur grapheme-to-phoneme [28] model, and the pronunciation dictionary that was used to train the models. As an added capability, this script can generate English phone time alignments with stress markings using models trained without stress markings. This is accomplished by first aligning the data with the unstressed models and then looking up the pronunciation of each word in a second dictionary that includes stress markings. For example, the dictionary used to train the models pronounces *cantaloupe* as /k ae n t ah l ow p/, and the dictionary with stress markings

¹⁵ Available at: <http://www.cstr.ed.ac.uk/projects/festival>

¹⁶ Available at: <http://www.portaudio.com>

¹⁷ Available at: <http://zeromq.org>

pronounces *cantaloupe* as /k ae1 n t ah0 l ow2 p/; the label 0 indicates no stress, 1 indicates primary stress, and 2 indicates secondary stress. Using the provided models we can obtain the time alignment for /k ae n t ah l ow p/ and then simply look up the stress labels *ae1*, *ah0*, and *ow2* in the second dictionary. A default stress marking of 0 is applied to all vowels if the unstressed pronunciation of a word does not exist in the stressed dictionary.

2.3.4. English Voices

This section describes the English voices that were created and discusses several experiments that were performed to compare the effects of different training data and models types. The first experiment compared voices trained on the CMU Arctic¹⁸ and Fisher corpora. Whereas the CMU Arctic corpus includes read speech recorded at a 16 kHz sampling rate with a close-talking microphone, the Fisher corpus consists of conversational telephone speech recorded at an 8 kHz sampling rate. Thus these corpora differ in speaking style (read vs. conversational), sampling rate (16 kHz vs. 8 kHz), and channel (close-talking microphone vs. telephone). The CMU Arctic corpus was downsampled to 8 kHz in order to match the Fisher corpus, and TTS systems were trained on a single speaker from each corpus. Each system used triphone labels with stress markings. Both voices yielded intelligible speech, although the voice trained on the Fisher corpus was substantially noisier.

The second experiment investigated four methods for improving the quality of an English TTS system. The baseline system was trained on a single speaker from the CMU Arctic corpus and used triphone labels. First, word initial position and syllable stress features were extracted using Festival and appended to the triphone labels. Second, the context window that is used when computing differential coefficients was expanded from ± 1 to ± 4 features. Third, whole word models were used for high frequency words. Fourth, an average voice model was trained on the WSJ corpus and then adapted using speech from the CMU Arctic corpus. In the author's opinion, the first method yielded a small improvement in voice quality and methods 2-4 yielded similar voice quality to the baseline system.

¹⁸ Available at: http://festvox.org/cmu_arctic

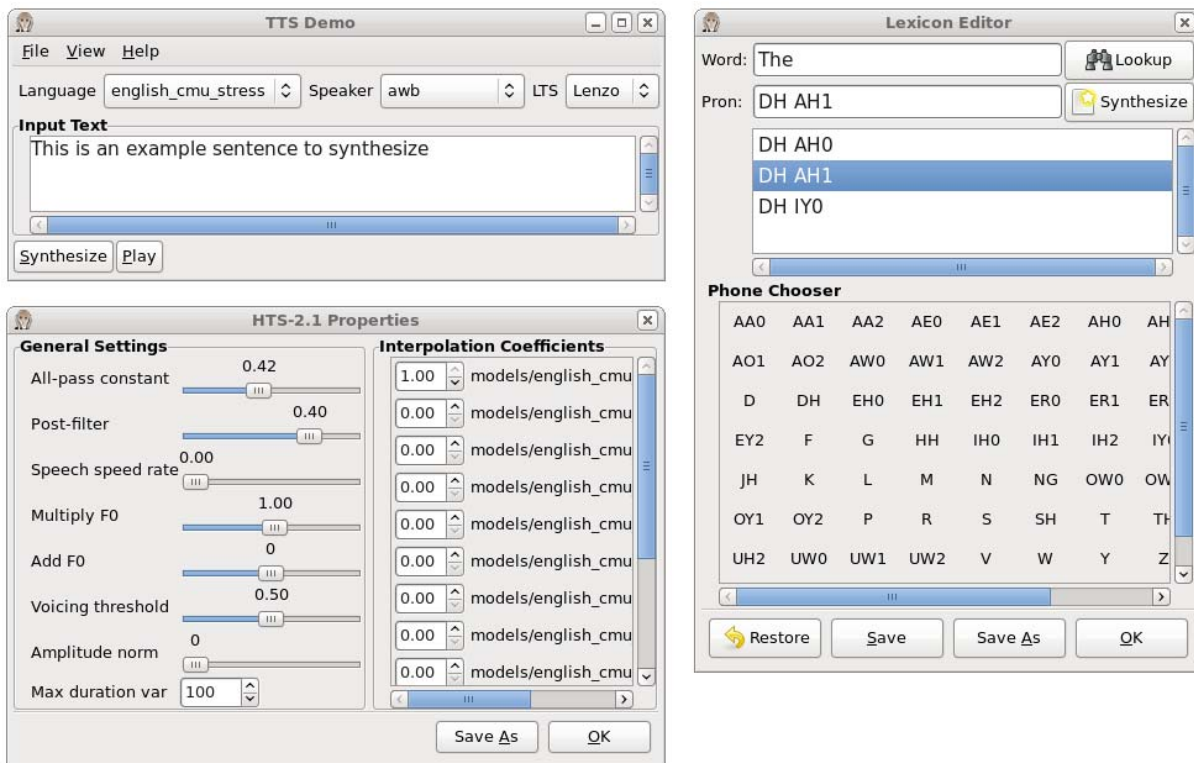


Figure 6: Text-to-Speech GUI



Figure 7: Audio Recorder GUI

The third experiment evaluated feature normalization. The baseline system was created by training an average voice model on three speakers, and then adapting this model to fit the characteristics of a fourth speaker. The models used triphone labels with stress markings and all speakers were chosen from the WSJ corpus. Feature normalization was applied by first normalizing the Mel-Cepstrum and Log F0 features for each speaker to zero mean and unit variance. Next, each feature vector was scaled so that global variance of the normalized features equaled the global variance of the un-normalized features. A second system was then trained on the normalized features. During synthesis, the generated Mel-Cepstrum and Log F0 coefficients were re-scaled using the target speaker's mean and variance. Overall, applying feature normalization yielded similar voice quality.

The fourth experiment compared average voice models trained with varying amounts of data per speaker. Three average voice models were trained on the same seven speakers, and then adapted to fit the characteristics of three other speakers. The models used triphone labels with stress markings and all speakers were chosen from the WSJ corpus. The average voice models were created using (1) 400 utterances per speaker, (2) 1240 utterances per speaker, and (3) up to 1240 utterances per speaker, excluding any utterances with word fragments, mispronunciations, or noise events. Increasing the number of utterances from 400 to 1240 per speaker yielded an improvement in voice quality, while methods 2 and 3 yielded similar results.

The fifth experiment compared models trained on triphone labels and models trained on quinphone labels with syllable, accent, stress, part-of-speech, word, and phrase information. First, an HMM speech recognition system was trained on 152 hours of speech produced by 358 speakers from the WSJ corpus. This system was developed with HTK and the Festival pronunciation dictionary. Phonemes were modeled using state clustered across word triphones, and the final HMM set included 5500 shared states with an average of 12 mixtures per state. The models were trained using MLE. The feature set was the same as described in Section 2.2.1. These models were then used to generate phone time alignments for each utterance from the WSJ corpus. Next, the Festival text analyzer was evaluated on each utterance and the labels were converted to HTS format. A total of four TTS systems were created using this label set: one voice was trained on 3.6 hours of speech from a single speaker, and three other voices were created by adapting an average voice model using 12 minutes of speech from each speaker. Compared to triphone labels, there was an improvement in voice quality for all systems; the most substantial improvement was obtained with the speaker dependent system trained on 3.6 hours of speech.

Lastly, TTS systems were created on data recorded in the SCREAM Laboratory. Two participants recorded 1132 prompts from the CMU Arctic corpus using a monotone speaking style, and then re-recorded the same set of prompts using an intonated or emphasized speaking style. All speech data were recorded at a 44.1 kHz sampling rate with an AKG C420 condenser microphone. The speech data was first down-sampled to a 16 kHz sampling rate. Phone time alignments were generated for each utterance using the WSJ speech recognizer, and quinphone labels with syllable, accent, stress, part-of-speech, word, and phrase information were extracted using Festival. Next, speaker-dependent TTS systems were trained for each speaking style. The intonated or emphasized speaking style yielded the best voice quality. Additional systems were created on the source data that was recorded at a 44.1 kHz sampling rate, and the speech data down-sampled to 22.05 kHz. The voices trained on the 22.05 kHz speech data yielded the best voice quality; however, all systems used 24th order Mel-Cepstrum coefficients and the optimal order varies with sampling rate.

2.3.5. Iraqi Voices

TTS systems were developed on Iraqi data from the TRANSTAC corpus. An average voice model was trained on 38 hours of speech produced by 173 speakers, and then adapted to fit the characteristics of 18 speakers. All of the speakers were male, and each adaptation speaker included 6–30 minutes of speech. Speaker dependent systems were also created for four of the adaptation speakers that had each recorded 30 minutes of speech. All systems were developed using triphone labels. Overall, the amount of adaptation data was not directly related to voice quality; other factors such as the average pitch of the speaker appear to be more important.

Compared to the speaker dependent models, the adapted systems yielded an improvement in voice quality.

2.3.6. Mandarin Voices

This section describes the Mandarin TTS systems that were created on the GlobalPhone corpus [29]. Prior to system development, the corpus transcripts were segmented into words using the Linguistic Data Consortium Chinese segmenter and digits were converted to Chinese characters. Next, phone time alignments were generated using a SONIC speech recognizer that was previously developed on the 1997 Broadcast News Speech corpus [30]. An average voice model was trained on 10 hours of speech produced by 53 male speakers, and then adapted using 2–16 minutes of speech produced by four different speakers. The models used triphone labels with tone markings.

One problem with the resulting voices was that audio clipping sometimes occurred when synthesizing speech with `hts_engine`. It was found that the Mel log spectrum approximation filter coefficients were the cause of problem. The following methods were identified for eliminating audio clipping: reducing the order of the Mel-Cepstrum analysis from 39 to 24, increasing the impulse response length applied during post filtering, or replacing the problematic Mel-Cepstrum pdfs in the adapted model with the corresponding pdfs from the average voice model. A script was developed to implement the third method and used to analyze 122 voices across five languages.

A second problem with the resulting voices was that the transitions between successive vowels were sometimes shorter in duration than expected and/or included sections of unvoiced speech. One reason for this phenomenon could be that there is a relatively small amount of training data for vowel sequences: vowels are followed by consonants 94% of the time in the training data. Removing decision tree questions that allowed consonants and vowels to be clustered into a single group (e.g., both consonants and vowels can be voiced) reduced the amount of unvoiced speech between successive vowels.

2.3.7. Spanish Voices

Spanish TTS systems were developed on the GlobalPhone corpus. An average voice model was trained on 5.8 hours of speech produced by 32 male speakers, and then adapted using 6–12 minutes of speech produced by 13 different speakers. Triphone labels were used all systems. A second set of models was developed on the same data using triphone labels with stress markings. The stress markings were automatically assigned using the following rules: words that end in a vowel, *n*, or *s*, are stressed on the next-to-last syllable, and words that end in other letters are stressed on the final syllable.¹⁹

2.3.8. Automation of Speaker Adaptation

Speaker adaptation using the HTS is time consuming, difficult, and poorly documented. In the process of attempting to document the steps required to create an adapted vocal model, it was determined that almost all of the steps could be automated. An adaptation script was created which reduces the entire process to a single step. The script reads its configuration values directly from the existing model, checks for data format problems, and runs ‘sanity checks’ on

¹⁹ http://spanish.about.com/od/spanishpronunciation/a/stress_accent.htm

any user supplied values. It can also use mixed formats for input data (WAV, MP3, or raw). This drastically reduces the chance of user error.

2.3.9. STRAIGHT Evaluation

The STRAIGHT synthesis system is built on the HTS and aims to increase the vocal quality of synthesized speech. The system was used to create vocal models for English, Urdu, Farsi, and Pashto. The vocal quality of the English voice was slightly better than with the standard HTS, but the other languages were far worse. The discrepancy in vocal quality was attributed to the differences in quality and length of the input data. The input data for English was high quality and each speaker had 1,132 samples. The input data for the other languages was not as clear and contained far fewer samples. It is possible that the vocal quality could be improved with a better set of input data.

2.3.10. Summary

This section summarizes the tools and models that were created for TTS. First, a previously developed text-to-speech GUI was ported from the Tcl/Tk programming language to Perl GTK. Functionality was added to perform English text analysis using Festival, generate pronunciations for unknown words using Kevin Lenzo's letter-to-sound program, save the generated speech to a file, and change the font size of the input text. Short pause models were added to each of the 151 voices available in the GUI. Second, an audio recorder GUI was developed for recording a speech database. Third, a Perl script was developed for automatically deriving phone time alignments. Support was provided for generating English phone time alignments with stress markings using models trained without stress markings.

A total of six experiments were performed using English TTS systems. First, voices were developed on the CMU Arctic and Fisher corpora. Next, four methods were investigated for improving the quality of a voice trained on the CMU Arctic corpus: (1) incorporating word initial position and syllable stress features, (2) expanding the context window used when computing differential coefficients, (3) using whole word models for high frequency words, and (4) adapting an average voice model trained on the WSJ corpus. Third, feature normalization was applied to a system trained on three speakers from the WSJ corpus. Fourth, average voice models were trained on seven speakers from the WSJ corpus using varying amounts of data per speaker. Fifth, models were compared using triphone labels and quinphone labels with additional contextual features that were extracted using Festival. Lastly, speech data were recorded in the SCREAM Laboratory and systems were developed using different speaking styles and sampling rates. Overall it was found that including additional contextual features in the labels, increasing the amount of training data, and using speech data that was recorded with an emphasized or intonated speaking style yielded improvements in voice quality.

Iraqi TTS systems were developed on the TRANSTAC corpus, and Mandarin and Spanish TTS systems were developed on the GlobalPhone corpus. Three methods were identified for eliminating audio clipping from synthesized speech. In addition, it was discovered that removing decision tree questions that include both constants and vowels reduced the amount of unvoiced speech between successive Mandarin vowels.

An automation system was created to reduce the effort necessary to create an adapted vocal model. It also allows the use of more audio formats for input data, eliminating the need to convert everything to raw audio.

HMM-based synthesis models were created using STRAIGHT for Pashto, Farsi, and English. The English model was created using 5 hours of speech from 5 speakers, the Pashto models were created using 1.8 hours of speech from 10 speakers, and the Farsi models were created using 9.5 hours of speech from 85 speakers.

2.3.11. Recommendations for Future Work

An improvement in voice quality was obtained on English by using quinphone labels with syllable, accent, stress, part-of-speech, word, and phrase information. Thus it may be beneficial to incorporate additional contextual features into the Iraqi, Mandarin, and Spanish systems. This would require the development of a text analyzer for each language.

English TTS systems were trained on speech data with a 44.1 kHz, 22.05 kHz, and 16 kHz sampling rate. All systems used 24th order Mel-Cepstrum coefficients. It would be interesting to retrain these systems using 16th to 39th order Mel-Cepstrum coefficients in order to evaluate the relationship between sampling rate and analysis order.

One problem with the Mandarin voices was that the transitions between successive vowels were sometimes shorter in duration than expected and/or included sections of unvoiced speech. This may be improved by using more training data or investigating additional decision tree questions. Another option would be to force a minimum duration and voiced speech when synthesizing consecutive vowels.

The generation of HMM-based TTS systems is still overly complex. It would be useful to create a more modular system for both training and adaptation, where configuration parameters are kept in a centralized place and input from the user is kept to a minimum. This could be accomplished by using a wrapper around the current HTS system. With the recent release of GPGPU cards with higher memory capacity, previously disregarded approaches to program acceleration may now be possible and should be investigated.

2.4 Laboratory Corpora Support

The quality of systems used in HLT is heavily dependent upon the quantity and quality of data used for model training, so the SCREAM Laboratory is constantly acquiring speech and language data and pre-processing data to improve quality. Section 2.4.1. discusses the Sada-e-Azadi Corpus, Section 2.4.2. covers the JRC Aquis Corpus, and Section 2.4.3. mentions the SETimes Corpus.

2.4.1. Sada-e-Azadi Corpus

Sada-e-Azadi²⁰ is a publication and website published by International Security Assistance Force (ISAF) in Afghanistan. Articles are published in Dari, Pashto, and English making them a valuable resource for document parallel text. While a portion of this work was performed over the duration of ICER Task Order 14 [7], the initial scripts were conceived under this task order as well as ongoing collection of new data.

²⁰ <http://www.sada-e-azadi.net>

Harvesting: Harvesting articles from the website involves manipulating Uniform Resource Locator's (URL's) in the site's Joomla content management system. Once the URL scheme had been decoded, access to the individual articles on a numeric article ID basis is possible. The python BeautifulSoup library allows us to extract the story body from the web page with a minimum of extraneous HTML formatting. Harvesting the articles by numeric ID allows us to easily retrieve the article in each of the three languages.

The format of the articles frequently changes on the Sada website, so tweaks to specifics of the harvesting process are necessary at times. To date, there have been approximately four major page format changes to the website.

Processing: After the pages are harvested from the website and have been moved into the lab network, additional processing is needed to transform the articles into parallel text suitable for training an SMT system. The first step is to examine the article triplets to ensure article version is in the correct language using a language classifier (TexCat). If any of the documents languages don't agree, then that triplet is not included in the final set of documents used to build the corpus.

Once the articles have been properly classified, any additional HTML tags that may have survived the article harvesting process are removed. Next, the comment fields are removed if present. Each version of the article share the same comments section and commenters do not use a uniform language while commenting.

Then the documents are processed by a sentence segmenter (NLTK's Punkt segmenter augmented with Arabic punctuation characters) to ensure that each sentence in the document is on a separate line. This is done to make the job easier for the sentence aligner. The sentence aligner, Champollion, is run on each language pair (English-Dari, English-Pashto) to determine a series of alignments for each document to create parallel text.

Once the alignment files have been generated, the documents are processed with the alignments and parallel data is produced, which is used to train SMT systems.

A refined version of the processing chain was introduced to both automate and speed up the cleaning and production of parallel data by reorganizing the data in such a manner that the process can be run in parallel on the lab's Open Grid Scheduler (OGS, formerly Sun Grid Engine) cluster.

Current Harvest Statistics: As of January 2014, the Sada-e-Azadi corpus consists of:

- 8278 document triplets (a document exists in English, Dari, and Pashto form)
- 103085 parallel lines English-Dari
- 108162 parallel lines English-Pashto

A corpus with 100,000 lines of parallel text is considered adequate to train a viable MT system.

2.4.2. JRC Aquis Corpus

The JRC Aquis²¹ corpus was collected by the European Commission and consists of European Commission proceedings.

²¹ <http://ipsc.jrc.ec.europa.eu/index.php?id=198>

This corpus uses the HunAlign sentence aligner to produce parallel data. Initial experiments were run to examine system performance on this data. The HunAlign sentence aligner was also compared to the Champollion sentence aligner used for other language pairs (Sada-e-Azadi corpus and others).

Data was extracted from portions of this corpus in order to supplement training data for MT systems. Additional post processing was performed to address any ongoing data formatting issues while training systems.

2.4.3. SETimes Corpus

The SETimes corpus is a collection of parallel text harvested from the SETimes website²². The corpus was extracted and initial experiments were run in order to support a distortion limit prediction metric, specifically Romanian-English which is not a commonly used language pair.

2.4.4. Summary

The primary corpus harvesting efforts were focused on the continued production of the Sada-e-Azadi corpus. Other corpora were downloaded in order to experiment on novel language pairs or to augment training data in other MT systems. Quality data is essential for training quality MT systems; the more data the better. Adding additional sources while maintaining existing ones is paramount.

2.4.5. Recommendations for Future Work

Work described in this section is being continued on other ICER Task Orders. Since Statistical MT and associated tasks are intensely data driven, it is appropriate to continue the harvesting and processing of data, especially in less common language pairs of interest.

2.5 Laboratory System Administration Support

Due to the computational and data storage intensive HLT research performed by the SCREAM Laboratory, there is a continuous need for System Administration support and Information Technology investment to maintain the computational efficacy of the network.

The SCREAM Laboratory network is comprised of high-performance workstations, high-performance rack-mounted computational nodes, and various servers including numerous high-capacity storage arrays, MT servers, web servers, database servers, backup servers, authentication servers, software and hardware inventory servers, centralized configuration servers, performance monitors, etc.

Utilizing OGS to manage the distributed execution of numerous data processing tasks from multiple users, the network is frequently processing data around the clock.

Under the ICER contract, system administration, software maintenance and upgrades, and hardware maintenance and upgrades for the SCREAM Laboratory, are shared among multiple task orders. In many cases, non-trivial system administration tasks were split between different task orders. While, some significant or otherwise interesting tasks are described below, they may also appear in reports for other task orders as the work was split between multiple task orders.

²² <http://www.setimes.com>

Comprehensive coverage of system administration tasks, maintenance, and upgrades is beyond the scope of this document and many frequent and/or routine system administration tasks, such as routine system maintenance, backups, system repair, system troubleshooting, and user support, also accomplished under this task order may not be listed.

2.5.1. Ethernet Infrastructure Upgrade

The SCREAM Laboratory network's port capacity, using 3 bridged 48-port Nortel gigabit Ethernet switches for a total of 144 ports, had been exceeded. Out of necessity, an additional switch was connected to the network in a non-optimal fashion using standard Ethernet cables. Although used for lower priority traffic, this connection increased the possibility of network performance degradation. Additionally, Nortel Networks Corporation filed for bankruptcy in January 2009 and ceased operations in June 2009. Finding, using, and supporting additional Nortel switches that could extend the existing network in an optimal fashion via cascade cables was determined to be unfeasible.

A decision was made to upgrade the network infrastructure. After a reasonable amount of market research, a solution was selected that could use the existing copper CAT 5/6 Ethernet cables with RJ-45 connectors and simultaneously support existing gigabit Ethernet connections and new 10 gigabit connections. The solution (purchased under a different task order) provides 240 10 gigabit Ethernet ports and consists of 5 48-port leaf switches and two spine switches from Arista Networks²³.

Each leaf switch connects to each spine switch with 2 x 40GB connections, providing 160GB of bandwidth to the spines. To increase available ports, two additional leaf switches could be added to provide 336 ports. If two additional spine switches were purchased, 15 leaf switches could be supported for a total of 720 ports.

After a few months of testing the Multi-Chassis Link Aggregation (MLAG) configuration and throttling between 10 gigabit and 1 gigabit connections, the switches were migrated to active use in January 2013.

²³ <http://www.aristanetworks.com>

3.0 CONCLUSIONS AND RECOMMENDATIONS

This document summarized work completed by SRA International, Inc. during the period 08 October 2008 to 31 March 2014.

The MT efforts reported here focus on the reduction of variation in the input. In general, Phrase-Based MT has trouble processing infrequent variant forms. Variations that occur with high frequency can generally be translated, since there is sufficient training data for each variant, but infrequent variants may remain untranslated.

Measuring variation is important, because normalization reduces information, and should only be applied when the variations cannot be handled by the MT system. Frequency-based techniques help the system by targeting normalization efforts to unknown or infrequent variants, and by ensuring that the resulting normalized forms are found in the data. Lattices retain information that is lost when using a single-best variant, while requiring less storage space than listing all the variants independently.

Various methods were incorporated allowing the Lab's translation services to communicate with each other. Scripts were developed to take advantage of the CyberTrans API that allowed for simple SOAP/PHP single line translations. This technology was then adopted into a web-based application to translate documents containing hundreds of thousands of lines of text.

An Experiment Reader was developed to manage the scores, statistics, dates and configuration files generated with each MT experiment. It might be beneficial to continue development of the Experiment Reader to allow for more fine-tuned viewing of the scoring and administrative functions. Currently iBLEU has to be directed to the hypothesis and reference files and manually sent to the scoring process. It would be an improvement to enable dynamic formatting of the hypothesis and reference files that are automatically created and sent off for immediate results.

GPGPU cards with higher memory capacity may allow data at our scales to be processed, especially with respect to neural networks. In addition, accelerated processing units from AMD that have shared memory space between the CPU and the GPU may provide a significant boost in performance without data transfer overhead.

Additional information about MT work performed under this Task Order can be found in Section 2.1.27 and the corresponding recommendations for future work in Section 2.1.28.

A script was developed for converting HTK models to Sphinx-4 ASCII format. As mentioned in Section 2.2.1, this script does not currently support feature transforms; it would be useful if transforms were supported since they are common in HMM systems. Also, Sphinx-4 does not currently support models with varying number of mixture components per state. This should be modified because HMMs with varying number of mixtures per state can yield an improvement in system performance.

WERs of 15.3% and 13.5% were obtained on the TRANSTAC Dari and Pashto corpus. This may indicate that there are errors in the transcripts. It would be beneficial to correct the transcripts so that more data can be used for training systems.

The best system on Callhome Mandarin yielded a 60.8% CER. Recall that the training partition for Callhome Mandarin only includes 12 hours of speech. The performance of this system may be improved by incorporating additional training data, or using a different modeling approach.

Possible suggestions include hybrid Deep Neural Network HMMs [23] or Recurrent Neural Network LMs [24]. The English systems may also benefit from these modeling techniques.

For additional information about ASR, the summary can be found in Section 2.2.5.

An improvement in voice quality was obtained on English by using quinphone labels with syllable, accent, stress, part-of-speech, word, and phrase information. Thus it may be beneficial to incorporate additional contextual features into the Iraqi, Mandarin, and Spanish systems. This would require the development of a text analyzer for each language.

English TTS systems were trained on speech data with a 44.1 kHz, 22.05 kHz, and 16 kHz sampling rate. All systems used 24th order Mel-Cepstrum coefficients. It would be interesting to retrain these systems using 16th to 39th order Mel-Cepstrum coefficients in order to evaluate the relationship between sampling rate and analysis order.

One problem with the Mandarin voices was that the transitions between successive vowels were sometimes shorter in duration than expected and/or included sections of unvoiced speech. This may be improved by using more training data or investigating additional decision tree questions. Another option would be to force a minimum duration and voiced speech when synthesizing consecutive vowels.

The generation of HMM-based TTS systems is still overly complex. It would be useful to create a more modular system for both training and adaptation, where configuration parameters are kept in a centralized place and input from the user is kept to a minimum. This could be accomplished by using a wrapper around the current HTS system. With the recent release of GPGPU cards with higher memory capacity, previously disregarded approaches to program acceleration may now be possible and should be investigated.

For additional information about TTS, the summary is in Section 2.3.10.

The primary corpus harvesting efforts were focused on the continued production of the Sada-e-Azadi corpus. Other corpora were downloaded in order to experiment on novel language pairs or to augment training data in other MT systems. Quality data is a necessity for training quality MT systems; the more data the better. Adding additional sources while maintaining existing ones is paramount.

Since Statistical MT and associated tasks are intensely data-driven, it is appropriate to continue the harvesting and processing of data, especially in less-common language pairs of interest.

4.0 REFERENCES

1. L. Schwartz et. al., “Static Offline Interpolation of Very Large Language Models with Renormalization of Backoff,” Air Force Research Laboratory, Unpublished manuscript, 2014.
2. J. Gwinnup, T. Anderson, “Democracy in Word Alignment,” SRA International, Unpublished manuscript, 2014.
3. Atkinson, Kevin. VARCON (Variant Conversion).
<http://wordlist.sourceforge.net/varcon-readme>
4. Linguistic Data Consortium. Less Commonly Taught Languages Urdu Corpus.
<http://projects.ldc.upenn.edu/LCTL>
5. Humayoun, Muhammad. Urdu morphological analyzer <http://www.lama.univ-savoie.fr/~humayoun/UrduMorph/>
6. XTAG morphology database. <http://www.cis.upenn.edu/~xtag/swrelease.html>
7. J. Gwinnup, K. Young, B. Ore, S. Thorn, D. Hoeferlin, D. Snyder, “Speech and Language Translation for Intelligence Community (SLTIC) Final Report,” SRA International. AFRL-RH-WP-TR-2013-0010, 2013.
8. Callison-Burch, Chris. <http://www.cs.jhu.edu/~ccb/howto-extract-paraphrases.html>
9. Bikel, Dan. Bikel parser. <http://www.cis.upenn.edu/~dbikel/software.html>
10. K. Young, J. Gwinnup, B. Ore, M. Hutt, S. Thorn, D. Hoeferlin, J. Cress, “Speech and Language Translation (SALT),” SRA International, Final Report. AFRL-RH-WP-TR-2012-0199, 2012.
11. A. Mauser, S. Hasan, and H. Ney, “Extending Statistical Machine Translation with Discriminative and Trigger-Based Lexicon Models,” in *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, Singapore, 2009, pp. 210-218.
12. Koehn, Philipp. Europarl: A Parallel Corpus for Statistical Machine Translation. [Online] 2005. <http://homepages.inf.ed.ac.uk/pkoehn/publications/europarl-mtsummit05.pdf>.
13. S. Young et al., “The HTK Book,” Cambridge University Engineering Department, 2006. Available: <http://htk.eng.cam.ac.uk>.
14. W. Walker et al., “Sphinx-4: A Flexible Open Source Framework for Speech Recognition,” Sun Microsystems, Tech. Rep. TR2004-0811, 2004.
15. J. Garofalo et al., “CSR-I (WSJ0) Complete,” Linguistic Data Consortium, 1993. Available: <https://www.ldc.upenn.edu>.
16. J. Garofalo et al., “CSR-II (WSJ1) Complete,” Linguistic Data Consortium, 1994. Available: <https://www.ldc.upenn.edu>.
17. A. Stolcke, “SRILM—An Extensible Language Modeling Toolkit,” in *Proc. of the International Conference on Spoken Language Processing (ICSLP)*, Denver, Colorado, USA, 2002, pp. 901–904.
18. A. Canavan and G. Zpperlen, “CALLHOME Mandarin Chinese Speech,” Linguistic Data Consortium, 1996. Available: <https://www.ldc.upenn.edu>.
19. C. J. Chen et al., “New Methods in Continuous Mandarin Speech Recognition,” in *Proc. of Eurospeech*, Rhodes, Greece, 1997.

20. C. Cieri *et al.*, “Fisher English Training Part 1,” Linguistic Data Consortium, 2004. Available: <https://www ldc upenn edu>.
21. C. Cieri *et al.*, “Fisher English Training Part 2,” Linguistic Data Consortium, 2005. Available: <https://www ldc upenn edu>.
22. J. Fiscus *et al.*, “2003 NIST Rich Transcription Evaluation Data,” Linguistic Data Consortium, 2007. Available: <https://www ldc upenn edu>.
23. G. Dahl *et al.*, “Context-Dependent Pre-trained Deep Neural Networks for Large Vocabulary Speech Recognition,” *IEEE Trans. Audio, Speech, and Language Processing*, vol. 20, no. 1, pp. 30–42, January 2012.
24. T. Mikolov *et al.*, “Recurrent Neural Network based Language Model,” in *Proc. of Interspeech*, Makuhari, Japan, 2010.
25. B. Ore, “Speech Recognition, Articulatory Feature Detection, and Speech Synthesis in Multiple Languages,” General Dynamics Advanced Information Systems, Interim Rep. AFRL-RH-WP-TR-2010-0067, Nov. 2009.
26. B. Pellom and K. Hacıoğlu, “SONIC: The University of Colorado Continuous Speech Recognizer,” University of Colorado, Tech. Rep. TR-CSLR-2001-01, May 2005.
27. A. Black, K. Lenzo, and V. Pagel, “Issues in Building General Letter to Sound Rules,” in *Proc. of the European Speech Communication Association (ESCA) Workshop on Speech Synthesis*, Jenolan Caves, Australia, 1998, pp. 77–80.
28. M. Bisani and H. Ney, “Joint-Sequence Models for Grapheme-to-Phoneme Conversion,” *Speech Communication*, vol. 50, no. 5, pp. 434–451, May 2008.
29. T. Schultz, “GlobalPhone: A Multilingual Speech and Text Database Developed at Karlsruhe University,” in *Proc. of the International Conference on Spoken Language Processing (ICSLP)*, Denver, Colorado, USA, 2002, pp. 345–348.
30. S. Huang *et al.*, “1997 Mandarin Broadcast News Speech (HUB4-NE),” Linguistic Data Consortium, 1999. Available: <https://www ldc upenn edu>.

5.0 LIST OF ACRONYMS & GLOSSARY

711 HPW	711 th Human Performance Wing
A3Metric	A collection of SCREAM Lab developed software tools to examine and modify results of word alignment
A3Threshold	A component to threshold A3-format word alignments based on word aligner score
ACL	Association for Computational Linguistics
AFRL	Air Force Research Laboratory
AMD	Advanced Micro Devices
AMTA	Association for Machine Translation in the Americas
AP5	A light morphological analysis procedure
API	Application Programming Interface
ARPA	Advanced Research Project Agency
ASCII	American Standard Code for Information Interchange
Aspell	A free open source spell checker
ASR	automatic speech recognition
BIA	Bilingual Alignment at the Phrase level word aligner
BLEU	Bilingual Evaluation Understudy, an algorithm for evaluating the quality of text which has been machine-translated from one natural language to another ²⁴ .
C	Consonantal [matching condition], a vowel-mapping method.
CAMT	Center for Applied Machine Translation
CB	Callison-Burch
CENTCOM	U.S. Central Command
CER	Character Error Rate
CMU	Carnegie Mellon University
CPAN	Comprehensive Perl Archive Network
CPU	Central Processing Unit
CQ	Consonant-qualified [matching condition], a vowel-mapping method
CSR	Continuous Speech Recognition
CV	Consonant-and-vowel [matching condition], a vowel-mapping method
DLI	Defense Language Institute
DoD	Department of Defense
DTIC	Defense Technical Information Center
EMNLP	Empirical Methods in Natural Language Processing
ESPS	Entropic Speech Processing System
Europarl	European Parliament

²⁴ <http://en.wikipedia.org/Wiki/BLEU>

FastAlign	A Simple, Fast, and Effective Reparameterization of IBM Model 2 word aligner
Fisher	An English corpus of conversational telephone speech
GATE	General Architecture for Text Engineering
GB	10 ⁹ bytes
GIZA	Training program that learns statistical translation models from bilingual corpora, part of The EGYPT Statistical Machine Translation Toolkit.
GOTS	Government Off the Shelf
GPGPU	General-purpose graphics processing unit
GPU	Graphics Processing Unit
GUI	Graphical User Interface
HPW	Human Performance Wing
HDecode	Cambridge University large vocabulary continuous speech recognizer
HLDA	Heteroscedastic Linear Discriminate Analysis
HLT	Human Language Technology
HMM	hidden markov model
HOTSPOT	A GOTS Language and Encoding Identification tool.
HTK	Hidden Markov Model Toolkit
HTML	HyperText Markup Language
HTS	HMM-Based Speech Synthesis System
IBM	International Business Machine
ICER	Information Operations Cyber Exploitation Research
ISAF	International Security Assistance Force
ISO	International Organization for Standardization
ITG	Inversion Transduction Grammar
IWSLT	International Workshop on Spoken Language Translation
JAPE	Java Annotation Patterns Engine
JAR	Java ARchive
Java	Java refers to a number of computer software products and specifications from Sun Microsystems that together provide a system for developing application software and deploying it in a cross-platform environment.
Joshua	an open source MT system developed at Johns Hopkins University
JRC	Joint Research Centre
KENLM	KenLM language modeling toolkit
LBFGS	Limited-memory Broyden-Fletcher-Goldfarb-Shanno
LCTL	Less Commonly Taught Languages
LDC	Linguistic Data Consortium
Lexapprox	Lexical Approximation
Linux	A Unix-like free and open source Operating System.

LM	language model
MADA	MADA is a full morphological tagger for Modern Standard Arabic developed by Nizar Habash, Owen Rambow and Ryan Roth.
Meteor	an automatic MT evaluation metric
Meteor/NEXT	an automatic MT evaluation metric based on Meteor
MFCC	Mel-Frequency Cepstral Coefficient
MIT	Massachusetts Institute of Technology
MIT/LL	Massachusetts Institute of Technology Lincoln Laboratory
MITLM	Massachusetts Institute of Technology Language Modeling Toolkit
MLAG	Multi-Chassis Link Aggregation
MLE	Maximum Likelihood Estimation
Morpha	Morphological Analyzer
Moses	A free software statistical MT engine
MP3	MPEG-2 Audio Layer III
MPE	Minimum Phone Error
MT	Machine translation (MT) is a sub-field of computational linguistics that investigates the use of computer software to translate text or speech from one natural language to another.
MT08	NIST 2008 Open Machine Translation Evaluation (MT08)
MT09	NIST 2009 Open Machine Translation Evaluation (MT09)
MT Eval	Machine Translation Evaluation
MultiBLEU	An additional MT scoring metric based on BLEU
MySQL	MySQL is a relational database management system (RDBMS) that runs as a server providing multi-user access to a number of databases.
MXPOST	Maximum Entropy Part-Of-Speech Tagger
NAIST	Nana Institute of Science & Technology
N-Best	A list of the top N sentences based on some score
NE	Named Entity
NEXGEN	Next-Generation Information Operations and Cyber Influence Capabilities
NILE	A word alignment software package
NIST	National Institute of Standards and Technology
NLP	natural language processing
NLTK	Natural Language Toolkit ²⁵
NPLM	Neural Probabilistic Language Model
OGS	Open Grid Scheduler, an open-source project based on SGE
OOV	Out Of Vocabulary
PDF	Portable Document Format (Adobe)

²⁵ <http://nltk.org>

Perl	Perl is a high-level, general-purpose, interpreted, dynamic programming language.
PHP	PHP: Hypertext Preprocessor is a widely used, general-purpose scripting language that was originally designed for web development to produce dynamic web pages.
PIAlign	Phrasal ITG word aligner
PLP	Perceptual Linear Prediction
PPDB	Paraphrase Database
POS	Part-of-Speech
PostCAT	Posterior Constrained Alignment Toolkit
RegAlign	Non-Deficient Model 3/4 Word Aligner
RevP	Reverse Palladius
RNNLM	Recurrent Neural Network Language Model
RPC	Remote Procedure Call
S2S	Speech-to-Speech
SAMT	Syntax-Augmented Machine Translation
SCREAM	Speech and Communication Research, Engineering, Analysis, and Modeling
SGE	Sun (now Oracle) Grid Engine, an open-source batch queuing system
SMT2	Statistical Machine Translation software created by MIT/LL and the SCREAM Laboratory that interfaces with Moses.
SLTS	Spoken Language Translation Systems
SOAP	Simple Object Access Protocol
SRILM	Stanford Research Institute LM toolkit
SRS	Speech Recognition System
Sphinx-4	An open source large vocabulary continuous speech recognition engine
Systran	A commercially available MT software system
STRAIGHT	A speech analysis, modification and synthesis system.
STL	Seed Translation Lexicon
SWAT	SCREAM Wikipedia Aided Translation
TED Talks	Technology, Entertainment, and Design is a global set of conferences owned by the private non-profit Sapling Foundation, formed to disseminate “ideas worth spreading” ²⁶ (http://www.ted.com/).
Thrax	An extractor for synchronous context-free grammars for use in MT
TRANSTAC	Spoken Language Communication and Translation System for Tactical Use
Treebank	A text corpus in which each sentence has been annotated with syntactic structure

²⁶ <http://en.wikipedia.org/wiki/Tedtalks>

Truecasing	Truecasing is the problem in NLP of determining proper capitalization of words when such information is unavailable ²⁷ .
TTS	Text-To-Speech Synthesis
URL	Uniform Resource Locator
UTF8	UTF-8 is a multibyte encoding that can represent any Unicode character.
UN	United Nations
VARCON	Variant Conversion
VOA	Voice of America
WAV	Waveform Audio File Format
WER	Word Error Rate
WMT	Workshop on statistical Machine Translation
WSDL	Web Services Description Language
WSJ	Wall Street Journal
XML	eXensible Markup Language
XML-RPC	XML Remote Procedure Call
XTAG	A lexicalized Tree Adjoining Grammar using an X-windows grammar development interface
ZeroMQ (ØMQ)	A high-performance asynchronous messaging library

²⁷ <http://en.wikipedia.org/wiki/Truecasing>

APPENDIX A - List of Russian MT Resources

A list of online resources for parallel text in Russian and English, as well as tools for Russian machine translation (compiled April 2013)

1. Websites with Multiple Documents, Most with Ongoing Postings
2. Single Large Documents
3. Minor Resources
4. Language Learning Sites
5. Corpora/Lexica
6. Russian Tokenizers, Morphological Analyzers, POS Taggers, and Parsers

1. WEBSITES WITH MULTIPLE DOCUMENTS, MOST WITH ONGOING POSTINGS

CENTCOM

<http://www.centcom.mil/>

This site is parallel throughout, in English, Russian, Arabic, Farsi, and Urdu. To switch languages, use the language tabs at the top of the page.

Russian listings:

<http://www.centcom.mil/ru/article/blog> (news articles)

<http://www.centcom.mil/ru/news/press-releases/> (press releases)

Select a particular article, then click the English tab to get its counterpart. For example:

<http://www.centcom.mil/ru/news/us-advisers-teach-self-defense-to-afghan-air-force-women>

<http://www.centcom.mil/news/us-advisers-teach-self-defense-to-afghan-air-force-women>

Note that the most recent articles are initially posted in English, but eventually get translated into Russian.

Count made in 2011, based on an average of 11 articles per page:

	News Listings	Articles Approx	Press Release Listings	Articles Approx
Urdu	18 pp	200	7+ pp	80
Farsi	42 pp	462	11 pp	121
Arabic	69 pp	759	18 pp	198
Russian	61 pp	671	13 pp	143
English	80 pp	880	40 pp	440

World Bank

<http://www.worldbank.org/>

<http://www.worldbank.org/eca/russian/>

Documents and reports in Russian: 768 articles.

Some are bilingual, some have separate versions in English and Russian.

<http://www-wds.worldbank.org/external/default/main?menuPK=64258548&pagePK=64187838&piPK=64187928&theSitePK=523679&function=BrowseFR&menuPK=64258548&siteName=WDS&conceptattcode=Russian~120705&pathtreeid=LANG&sortattcode=DOCDT+Desc>

For example, 68 pp, Rising food and energy prices in Europe and Central Asia.

http://www-wds.worldbank.org/external/default/main?menuPK=64187510&pagePK=64193027&piPK=64187937&theSitePK=523679&menuPK=64154159&searchMenuPK=64258548&theSitePK=523679&entityID=000333038_20120117221729&searchMenuPK=64258548&theSitePK=523679

News articles: over 1000 articles (62 pages of listings, with about 19 articles per page).

<http://www.worldbank.org/ru/news>

<http://web.worldbank.org/WBSITE/EXTERNAL/NEWS/0,,enableDHL:TRUE~menuPK:51062075~pagePK:64001221~piPK:51161268~theSitePK:4607,00.html>

News articles in Russian:

<http://web.worldbank.org/WBSITE/EXTERNAL/NEWS/0,,lang:120705~menuPK:64256856~pagePK:64254793~piPK:64256860~theSitePK:4607,00.html>

For example, The World Bank Country Partnership Strategy for 2012-2017: Public Consultations. Click on the Russian headline to bring it up:

<http://web.worldbank.org/WBSITE/EXTERNAL/NEWS/0,,contentMDK:23143205~menuPK:64256856~pagePK:34370~piPK:34424~theSitePK:4607,00.html>

Then click on the link in the upper right, "also available in English":

<http://web.worldbank.org/WBSITE/EXTERNAL/NEWS/0,,contentMDK:23143155~enableDHL:TRUE~menuPK:64256856~pagePK:34370~piPK:34424~theSitePK:4607,00.html>

Daniel Pipes Political Blog

www.danielpipes.org

<http://ru.danielpipes.org/art/year/all>

About 350 articles in Russian; these are not named in parallel, but the Russian pages include links to the English version near the top of the article, given as the English title.

US Embassy in Moscow

<http://moscow.usembassy.gov/>

Searching on Перевод English ("translation English") gives 90 results:

http://search.state.gov/search?q=%D0%9F%D0%B5%D1%80%D0%B5%D0%B2%D0%BE%D0%B4+English&client=emb_ru_moscow&output=xml_no_dtd&proxystylesheet=emb_ru_moscow&oe=UTF-8&ie=UTF-8&lr=lang_ru&filter=0&access=p&sort=date:D:L:d1&entsp=0&ud=1&exclude_apps=1&site=emb_eur_moscow&&&ip=68.177.49.70&entqr=0&num=100

Click on an article title to pull up the Russian article.

To get the English version, click on the link at the top of the article: Перевод:English

Council of Europe Treaties

117 documents in Russian, English, and French

http://conventions.coe.int/?pg=Treaty/Translations/Translations_rus.htm

Get the corresponding English versions by treaty number here:

<http://conventions.coe.int/Treaty/Commun/ListeTraites.asp?CM=8&CL=ENG>

TED Talks

<http://www.ted.com/translate/languages/ru>

United Nations

Enter a keyword search of the UN document database here:

<http://unbisnet.un.org:8080/ipac20/ipac.jsp?session=D332960C36962.682218&profile=bib&menu=search&submenu=subtab124&ts=1332960339078#focus>

Search by:

Field 1: Date of Publication: 2012

Databases: UN Publications

Language: Russian

Sorting: Publication Date (Ascending)

This returns documents that are available in Russian. The listing for the document includes links to all language versions. Since most of these are also available in English, this gives us an index of parallel documents.

For 2012, there were 19 documents. For 2011, there were 495 documents.

http://www-pub.iaea.org/books/IAEABooks/Advanced_Search/Results/russian-edition_0_0_Any_0_Any_Any_0

Also found 77 documents with "Russian Edition" in the title; the list gives links to other language editions.

US Holocaust Museum

About 30 articles translated into Russian.

<http://www.ushmm.org/>

List of the Russian articles:

<http://www.ushmm.org/museum/exhibit/focus/russian/>

The English versions are named in parallel, for example:

<http://www.ushmm.org/wlc/ru/article.php?ModuleId=10005143>

<http://www.ushmm.org/wlc/en/article.php?ModuleId=10005143>

2. SINGLE LARGE DOCUMENTS

Canadian Government Documents

Welcome to the United States: A Guide for New Immigrants, 92 pages

http://www.uscis.gov/files/nativedocuments/M-618_r.pdf

<http://www.uscis.gov/files/nativedocuments/M-618.pdf>

Welcome to Ontario, 47 pages

<http://www.citizenship.gov.on.ca/english/publications/docs/welcometoontario/Welcome-to-Ontario.russian.pdf>

<http://www.citizenship.gov.on.ca/english/publications/docs/welcometoontario/Welcome-to-Ontario.eng.pdf>

First Days in Ontario, 37 pages

http://www.ontarioimmigration.ca/stdprodconsume/groups/csc/@oipp/documents/document/oi_first_days_russian.pdf

http://www.ontarioimmigration.ca/stdprodconsume/groups/csc/@oipp/documents/document/oi_first_days_guide.pdf

Alone in Canada, 56 pages

http://www.camh.net/About_Addiction_Mental_Health/Mental_Health_Information/Alone_in_Canada/alone_in_canada_russian.pdf

http://www.camh.net/About_Addiction_Mental_Health/Mental_Health_Information/Alone_in_Canada/alone_in_canada.pdf

United States Government Documents

FEMA

http://www.fema.gov/pdf/assistance/process/help_after_disaster_russian.pdf 36 pp

http://www.fema.gov/pdf/assistance/process/help_after_disaster_english.pdf

Foreign Corrupt Practices Act

<http://www.justice.gov/criminal/fraud/fcpa/docs/fcpa-russia.pdf> 24 pp

<http://www.justice.gov/criminal/fraud/fcpa/docs/fcpa-english.pdf> 16 pp (different size fonts)

Federal Protections Against National Origin Discrimination

<http://www.justice.gov/crt/publications/natlgov-rus.pdf> 21 pp

<http://www.justice.gov/crt/publications/natlgov2.pdf> 16 pp

The Federal Court System in the United States: An Introduction for Judges and Judicial Administrators in Other Countries

http://www.fjc.gov/ijr/home.nsf/page/transl_mat#Russian

<http://www.uscourts.gov/uscourts/FederalCourts/Publications/russian.pdf> 77 pages

<http://www.uscourts.gov/uscourts/FederalCourts/Publications/English.pdf>

World Bank

Six articles that may be a good source of geographic entity names; but note that some contain just the front matter.

<http://wdronline.worldbank.org/worldbank/a/langtrans>

Select Russian in the drop down menu, click on article links to download pdf.

<http://wdronline.worldbank.org/worldbank/a/browsebytitle>

Find corresponding English articles by title.

World Development Report 2012 : Gender Equality and Development (67pp)

World Development Report 2011 : Conflict, Security, and Development (7 pp) [front matter only]

World Development Report 2009 : Reshaping Economic Geography (411 pp) [includes maps with geographic names; list of acronyms]

World Development Report 2007 : Development and the Next Generation (378 pp)

World Development Report 2006 : Equity and Development (311 pp)

World Development Report 2005 : A Better Investment Climate for Everyone (290 pp)

University Digests and Other Digests

The Current Digest of the (Post-)Soviet Press

<http://dlib.eastview.com/c>

Click on the title, *Current Digest*.

Weekly translated summaries of selected articles in Soviet and post-Soviet press. This can be used as partial index to articles in Russian.

World News Connection.

<http://library.stanford.edu/>

Click on Databases & Articles, then search for the title, *World News Connection*.

Full-text English language translations of selected foreign television and radio broadcasts, newspaper & journal articles, press releases, from 1996 on.

Columbia Newsblaster

<http://www.cs.columbia.edu/techreports/cucs-010-03.pdf>

Newsblaster Russian-English Clustering Performance Analysis, by Lawrence J. Leftin

Abstract: The Natural Language Group is developing a multi-language version of Columbia Newsblaster, a program that generates summaries of news articles collected from web sites. Newsblaster currently processes articles in Arabic, Japanese, Portuguese, Spanish, and Russian, as well as English. This report outlines the Russian language processing software, focusing on machine translation and document clustering. Russian-English clustering results are analyzed and indicate encouraging inter-language and intra-language performance.

WPS: What the Papers Say

http://www.wps.ru/e_index.html

Daily English language survey of the Russian media containing translations and summaries in full-text of the most important articles selected from major daily and weekly news sources, as well as transcripts of nightly television and radio news and public affairs programs; available by subscription as PDF files in full-text.

The Current digest of the Soviet and post-Soviet press

<http://library.duke.edu/metasearch/db/id/DUK02263> A selection of Russian-language press materials, carefully translated into English, 1949-present.

Universal database of Russian Social Sciences and Humanities Publications .

Access by subscription only.

Full-text electronic versions of the leading Russian social science and humanities journals. Provides coverage of current publications in the areas of: economics, international relations, public policy, sociology, public health, archaeology, history, and literary criticism.

Russian Library of Science

<http://www.springer.com/librarians/russian+library+of+science?SGWID=0-40748-0-0-0>

Springer. Over 200 Russian journals in English. Contains articles and papers from research institutes and scientific societies in Russia and the surrounding countries.

For example:

Biomedical Engineering is a translation of the peer-reviewed Russian journal *Meditsinskaya Tekhnika*. The Russian volume-year is published in English beginning in July.

Russian language information on *Meditsinskaya Tekhnika* is available at the following link:

<http://www.mtjournal.ru/>

3. MINOR RESOURCES

Literature

Various sites have put parts of Russian novels online in Russian and English.

<http://www.russianlessons.net/ebooks/show/1/1/7/both>

Part I of *Crime and Punishment*, paragraph aligned.

http://learningrussian.net/anna_karenina.php

Part of *Anna Karenina* and *Notes from the Underground*, paragraph aligned.

Note: displays in columns, but downloads as alternate paragraphs of Russian and English.

[English-Russian, Russian-English fiction](#); a small parallel corpus of English and Russian fiction from the 19th century.

<http://lib.ru/TRANSLATION/>

Russian Science Fiction in Translation: a dozen or more novels.

Note: versions may differ in terms of introductory material, but the main texts appear parallel.

<http://www.let.rug.nl/~houtzage/alicerussengpart1.html>

Alice in Wonderland (5 pages), Winnie the Pooh (4 pages), translated into Russian

Note: displays in parallel columns, by stanzas, numbered.

Health Brochures

Health Translations, Victoria, Australia. 334 Russian brochures (some links broken).

<http://healthtranslations.vic.gov.au/bhcv2/bhcht.nsf/PresentMultilingualResource?Open&x=&s=Russian>

Refugee Health Information Network: 146 brochures.

<http://www.rhin.org/Search.aspx?source=advancedSearch&LanguageID=57&FormatID=1>

Virginia Department of Health: 16 health brochures.

http://www.vdh.virginia.gov/CLAS_Act/languageresources/translatedvdh/lang_russian.htm

Medicare: 11 documents.

[http://www.medicare.gov/\(X\(1\)S\(sdpXoo4500xqw33icr1g0u2n\)\)/multilanguage.aspx?AspxAutoDetectCookieSupport=1#ru](http://www.medicare.gov/(X(1)S(sdpXoo4500xqw33icr1g0u2n))/multilanguage.aspx?AspxAutoDetectCookieSupport=1#ru)

Problem Gambling Institute of Ontario: 10 health brochures

<http://www.problemgambling.ca/EN/PGDocuments/Pages/TranslatedResourcesRussian.aspx>

4. LANGUAGE LEARNING SITES

Global Language Online Support System: 459 Russian lessons.

<http://gloss.dliflc.edu/>

SCOLA: 9 Russian lessons.

www.scola.org

Russian Learn

<http://russianlearn.com>

About 30 dual language articles, with Russian and English in columns, paragraph aligned; also vocabulary lists. Some articles are political, some are popular culture.

Russian Lessons

<http://www.russianlessons.net/articles/index.php>

19 dual-language articles, paragraph aligned, on introductory topics about Russia.

5. CORPORA/LEXICA

OPUS

<http://opus.lingfil.uu.se/>

Select en/ru.

Abstract: OPUS is a growing collection of translated texts from the web. In the OPUS project we try to convert and align free online data, to add linguistic annotation, and to provide the community with a publicly available parallel corpus. OPUS is based on open source products and the corpus is also delivered as an open content package. We used several tools to compile the current collection. All pre-processing is done automatically. No manual corrections have been carried out.

Please [cite the following article](#) if you use any part of the corpus in your own work:

Jörg Tiedemann, 2009, [News from OPUS - A Collection of Multilingual Parallel Corpora with Tools and Interfaces](#). In N. Nicolov and K. Bontcheva and G. Angelova and R. Mitkov (eds.) Recent Advances in Natural Language Processing (vol V), pages 237-248, John Benjamins, Amsterdam/Philadelphia

download resources:

corpus	doc's	sent's	src tokens	trg tokens	XCES/XML	Moses	TMX			Browse Files
OpenSubtitles 2011	6030	5,0M	39,7M	34,2M	[xces en ru]	[mooses]	[tmx]		[sample]	[xml/en] [xml/ru] [xml/en-ru]
KDE4	1454	0,2M	1,8M	1,4M	[xces en ru]	[mooses]	[tmx]	[query]	[sample]	[xml/en] [xml/ru]
PHP	3281	35,6k	0,5M	0,1M	[xces en ru]	[mooses]	[tmx]	[query]	[sample]	[xml/en] [xml/ru]
OpenSubtitles	28	24,9k	0,2M	0,2M	[xces en ru]	[mooses]	[tmx]	[query]	[sample]	[xml/en] [xml/ru] [xml/en-ru]
total	10793	5,2M	42,3M	35,9M	5,2M	4,9M	3,8M			

OPUS provides data from PHP, which was originally extracted from <http://se.php.net/download-docs.php>.

Abstract for PHP: The original documents are written in English and have been partly translated into 21 languages. The original manuals contain about 500,000 words. The amount of actually translated texts varies for different languages between 50,000 and 380,000 words. The corpus is rather noisy and may include parts from the English original in some of the translations. The corpus is tokenized and each language pair has been sentence aligned.

OPUS also provides Russian-English data from Tatoeba: <http://opus.lingfil.uu.se/Tatoeba.php>; see also separate listing, below.

Tatoeba

http://tatoeba.org/eng/sentences/show_all_in/rus/none/none/indifferent

The Tatoeba website lists 78K sentences for Russian.

Abstract: Tatoeba.org is a free collaborative online [database](#) of example sentences geared towards foreign language learners. Tatoeba focuses on [translation](#) of [complete sentences](#). In addition, the structure of the database and interface emphasize one-to-many relationships. Not only can a sentence can have multiple translations within a single language, but its translations into all languages are readily visible, as are indirect translations that involve a chain of stepwise links from one language to another. As of December 2012, Tatoeba's corpus has 2,000,000 sentences in 118 languages. A list of how many sentences there are in each language can be found on Tatoeba's [language statistics page](#).

Temporally Aligned English-Russian Corpus

<http://cogcomp.cs.illinois.edu/Data/Temporal/temporal.html>

Data: [complete.tar.gz](#)

Evaluation NE pairs: [evalpairs.txt](#)

Abstract: The corpus contains 2345 publicly available BBC new articles and their loose Russian translations collected over the period of 01/01/2001 to 10/05/2005 from news.bbc.co.uk and www.lenta.ru.

University of Leeds Corpora

<http://corpus.leeds.ac.uk/>

Centre for Translation Studies, University of Leeds, develops and hosts a range of large representative corpora in a variety of languages (including English, Arabic, Chinese, French, German, Italian, Japanese, Spanish, Polish and Russian).

Comparable Corpus of English and Russian News Texts

Originally the website was created for making [the query interface](#) to the comparable corpus of English and Russian news texts. The English corpus is based on a subset of the corpus of Reuters news, a collection of newswires from Reuters for one year from 1996-1997. The Russian corpus is based on articles from Izvestia, a national broadsheet newspaper, and covers the period from 2000 to 2001.

The use of the corpus is restricted for research purposes only. Because of the nature of our agreement with Reuters we have to monitor the users of their subcorpus. This requires free registration for interested users.

Russian Reference Corpus

About 50 million words from a variety of genres. The Russian Reference Corpus was also used as the basis for development of the frequency dictionary of modern Russian, its description and information for download is available from [a separate page](#).

<http://corpus.leeds.ac.uk/ruscorpora.html>

The page lists four corpora:

1. a pilot version of the Russian National Corpus (50 million words, a representative collection of various genres, see <http://ruscorpora.ru>, the mirror here is provided by courtesy of the Moscow team), the complete set of sources is available from [here](#);
2. the corpus of Russian newspapers (70 MW, consisting of several major Russian newspapers, 2001-2004);
3. the Russian Internet Corpus (160 MW, a snapshot of modern Russian language as used on the Internet; this is work in progress, which is similar to other [Internet Corpora](#) for Chinese, English and German);
4. a corpus of Russian fiction (1.5 MW; its morphosyntactic features have been manually disambiguated by a team led by Vladimir Plungyan), the complete set of sources is available from [here](#).

Russian National Corpus

Description given in Wikipedia:

The Russian National Corpus (English official name; the Russian name is Национальный корпус русского языка, lit. the National Corpus of the Russian language, but as the official English variant the Russian National Corpus is used) is a corpus of the Russian language that has been partially accessible through a query interface online since April 29, 2004. It is being created by the Institute of Russian language, Russian Academy of Sciences.

It currently contains about 350 million word forms that are automatically lemmatized and POS-/grammeme-tagged, i. e. all the possible morphological analyses for each orthographic form are ascribed to it. Lemmata, POS, grammatical items and their combinations are searchable.

Additionally, 6 million word forms are in the subcorpus with manually resolved homonymy.

The subcorpus with resolved morphological homonymy is also automatically accentuated. The whole corpus has a searchable tagging concerning lexical semantics (LS),[1] including morphosemantic POS subclasses (proper noun, reflexive pronoun etc.), LS characteristics proper (thematic class, causativity, evaluation), derivation (diminutive, adverb formed from adjective etc.).

The RNC includes also the following subcorpora:

- a treebank of syntactical dependencies (largely based on the Igor Mel'čuk's Meaning-Text Theory)
- English⇔Russian, German⇒Russian, Ukrainian⇔Russian and Belorussian⇔Russian parallel corpora;
- a large (100+ million words) separate corpus of modern newspapers (2001–2011);
- a corpus of Russian poetry, where the rhyming words and poetic prosody (including meter, stanzas etc.) is additionally tagged;
- a corpus of Russian dialects with specific dialect grammar tagging;
- a multimedia corpus with searchable tagged fragments of Russian-language movies;
- a corpus showing the history of Russian stress
- an educational subcorpus reflecting school standards.

All the texts have tags bearing metatextual information - the author, his/her birth date, creation date, text size, text genres (general fiction, detective story, newspaper article etc.); all these

categories are browsable and searchable separately. It is possible to define a user's subcorpus to search lemmata/POS-grammeme/semantic tags combinations only within this subset.

Descriptions given on the RNC website itself:

All the texts in the RNC are presented on this site online and are available for non-commercial scientific or educational use (Article 19 of the Russian Copyright Law). Currently no text on the site can be downloaded and/or read in full.

If you quote the contexts retrieved from the RNC please cite RNC as the source as well as the author of the text in question and the name of the text.

The Corpus currently is unavailable online. A 180-thousand-tokens random selection of sentences is available for downloading:

[download the sample
tagset](#)

Word lists by ngrams. Словоформы = word forms, топ-100 = top 100, 2-граммы = 2gram, zip-архив = zip archive)

Словоформы	zip-архив (5,5 Мб, обрезаны по частоте 3)	топ-100
2-граммы	zip-архив (39 Мб, обрезаны по частоте 3)	топ-100
3-граммы	zip-архив (31 Мб, обрезаны по частоте 3)	топ-100
4-граммы	zip-архив (44 Мб, обрезаны по частоте 2)	топ-100
5-граммы	zip-архив (28 Мб, обрезаны по частоте 2)	топ-100
6-граммы		топ

Glossaries

New York City Department of Education glossary of education terms

<http://schools.nyc.gov/NR/ronlyres/390C9547-8002-48C8-AD55-CA497E3728CE/45868/RussianGlossary.pdf>

187 pages, about 30 terms per page; terms include jargon (e.g., names of local committees) but also common words and phrases.

Eurasia Health

<http://www.eurasiahealth.org/index.jsp?lng=ru>

Healthsearchable glossary of health terms; cannot be downloaded.

6. RUSSIAN TOKENIZERS, MORPHOLOGICAL ANALYZERS, POS TAGGERS, AND PARSERS

Grammatical Note

Russian word order is not restricted to strict subject-verb-object order, but rather expresses semantic relationships. Typically, new information (theme) is presented before old information (rheme). Russian parsers often use dependency grammars, in which relationships between words are emphasized as opposed to the linear order and structure of the phrases. This can make more sense for a language with relatively free word order like Russian.

University of Leeds Survey of Russian MT Tools

<http://corpus.leeds.ac.uk/mocky/>

The University of Leeds website gives detailed information and links for a variety of Russian MT tools. Excerpts from this site are listed here for taggers, stemmers, and parsers.

Tagging Resources

You can download the following resources:

[russian.par.gz](#) - a parameter file to be used with TreeTagger, for using it you need a tokeniser and TreeTagger, available from <http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/>

[russian-tnt.tgz](#) - parameter files to be used with TnT, <http://www.coli.uni-saarland.de/~thorsten/tnt/>

[russian-svm.tgz](#) - parameter files to be used with SVMTagger, <http://www.lsi.upc.es/~nlp/SVMTool/>

[ru-table.tab](#) - a table mapping MSDs to a combination of POS categories (tab-separate file format);

[i-ru-sample.txt.gz](#) - a Russian corpus for tagging experiments. It is a 5mln word subset from the Russian Internet Corpus (Sharoff, 2006 in <http://wackybook.sslmit.unibo.it/>);

several tagged texts produced from the Russian sample by TreeTagger ([i-ru-sample-tt.out.gz](#)), TnT ([i-ru-sample-tnt.out.gz](#)) and SVMTagger (Model 0, [i-ru-sample-svm0.out.gz](#); Model 2, [i-ru-sample-svm2.out.gz](#); Model 5, [i-ru-sample-svm5.out.gz](#))

[russian-small.par.gz](#) - a parameter file for TreeTagger that uses a small tagset (only the basic POS tags are distinguished according to the model of <http://www.ruscorpora.ru/>)

All files use UTF-8 encoding. They have been trained on the disambiguated version of the Russian National Corpus (<http://www.ruscorpora.ru>), which cannot be made available because of copyright reasons. However, the tagged files from the Internet sample can provide a basis for further training. You are welcome to investigate problems in each tagged file or combine them (e.g., by majority voting) to produce a better training corpus.

Stemming Resources

The resources for TreeTagger can do lemmatisation by themselves, if you use -lemma for tagging (this is based on a form+pos-to-lemma lexicon). However, this mechanism does not help with lemmatisation of unknown word forms. Bart Jongejan and his colleagues from the Danish

[Center for Sprogteknologi](#) developed [CSTlemma](#), a tool that learns morphosyntactic rules from form+pos+lemma triples.

My [lemmatisation tool](#) is a wrapper around CSTlemma (which has to be downloaded and installed separately). The tool takes the output of TnT or TreeTagger, assigns lemmas from a dictionary in the TreeTagger format (which can be gzipped to save space) and uses cstlemma to guess unknown inflected forms (adjectives, nouns and verbs). A remark for guessed forms is left in the fourth column.

Russian Dependency Parsing

Statistical approaches can be also used to create parsers. In cooperation with Joakim Nivre I have created a [Russian dependency parser](#) for the [Malt Parser](#) (version 1.5). The training corpus for the parser was SynTagRus as developed by Igor Boguslavsky, Leonid Iomdin and their colleagues (<http://cl.iitp.ru/>). This means that the set of dependency relations is the same as the output of the ETAP parser, for an overview of the labels see the [Russian National Corpus](#) (in Russian).

[The script](#) for running the Russian Malt parser on a text encoded in UTF-8 is invoked as:

```
./russian-malt.sh <input >output
```

In [a competition of Russian dependency parsers in 2012](#) this simple parser produced fairly reliable results, ranking 3 out of 8 by the F-measure.

The parser has been applied to parse ruWac, a 2 billion word corpus of Russian (a representative snapshot of the Russian Web). The parsed file is available from [here](#) (warning, this downloads 9GB of compressed text).

To refer to the parser, please use:

Sharoff, S., Nivre, J. (2011) The proper place of men and machines in language technology: Processing Russian without any linguistic knowledge. *Proc. Dialogue 2011, Russian Conference on Computational Linguistics*. [PDF](#)

Other Russian MT Tools

OpenCorpora Tokenizer

<http://search.cpan.org/~ksuri/Lingua-RU-OpenCorpora-Tokenizer-0.01/lib/Lingua/RU/OpenCorpora/Tokenizer.pm>

Kimmo Morphological Parser

http://www.sil.org/computing/catalog/show_software.asp?id=33

PC-KIMMO is a morphological parser based on Kimmo Koskeniemi's two-level model of morphology. It is distributed by the Summer Institute of Linguistics.

<http://language.link.let.uu.nl/~lion/index.php>

Russian implementation

aot.ru POS tagger

http://packages.python.org/pymorphy/ref/Morph_UNIX.html

This is a program of morphological analysis (Russian, German, and English languages).

The Russian lexicon is based upon Zaliznyak's Dictionary.

Articles

Text segmentation in OpenCorpora [in Russian]

2012 International Conference on Computational Linguistics

<http://www.dialog-21.ru/en/digest/2012/?type=content>

Abstract: By segmentation we mean text tokenization and sentence splitting. By default most text processing pipelines start from these two jobs before any morphology or syntax. Moreover later analysis stages depend on the way tokenization and sentence splitting are performed. In OpenCorpora project we have done manual tokenization and sentence splitting works on about 600K textforms. A machine learning technique has been trained on this data and obtained results as well as segmentation standard and its motivation are described in this article.

Language modelling for Russian and English using words and classes

E.W.D Whittaker, , P.C Woodland

<http://www.sciencedirect.com/science/article/pii/S0885230802000475>

Abstract: This paper examines statistical language modelling of Russian and English in the context of automatic speech recognition. The characteristics of both a Russian and an English text corpus of similar composition are discussed with reference to the properties of both languages. In particular, it is shown that to achieve the same vocabulary coverage as a 65,000 word vocabulary for English, a 430,000 word vocabulary is required for Russian.

About the creation of a parallel bilingual corpora of web-publications

D.V. Lande and V.V. Zhygalo

<http://arxiv.org/ftp/arxiv/papers/0807/0807.0311.pdf>

Available electronic dictionaries were taken for Russian and Ukrainian languages (ispell with over 1.102 thousand word forms in the Ukrainian language and Zalizniak's dictionary which enumerates 93 392 words in a regular form).

Building a Web-based parallel corpus and filtering out machine translated text

Alexandra Antonova and Alexey Misyurev

<http://www.mt-archive.info/BUCC-2011-Antonova.pdf>

This 2011 article discusses building a Russian/English corpus by detecting parallel web documents; it describes how to determine if documents are parallel, and how to screen out machine-translated documents.

Russian morphology: An engineering approach

Andrei Mikheev and Liubov Liubushkina

<http://journals.cambridge.org/action/displayAbstract?fromPage=online&aid=1279148>

Morphological analysis, which is at the heart of the processing of natural language requires computationally effective morphological processors. In this paper an approach to the organization of an inflectional morphological model and its application for the Russian language

are described. This approach allows us to handle uniformly both general cases and exceptions, and requires extremely simple data structures and control mechanisms which can be easily implemented as a finite-state automata. The morphological processor described in this paper is fully implemented for a substantial subset of Russian (more than 1,500,000 word-tokens – 95,000 word paradigms) and provides an extensive list of morpho-syntactic features together with stress positions for words utilized in its lexicon. Special dictionary management tools were built for browsing, debugging and extension of the lexicon. The actual implementation was done in C and C++, and the system is available for the MS-DOS, MS-Windows and UNIX platforms.

Development of a Dependency Treebank for Russian and its Possible Applications in NLP

Boguslavsky et al.

<http://www.google.com/url?sa=t&rct=j&q=russian%20treebank&source=web&cd=5&cad=rja&ved=0CFMQFjAE&url=http%3A%2F%2Fciteseerx.ist.psu.edu%2Fviewdoc%2Fdownload%3Fdoi%3D10.1.1.102.9324%26rep%3Drep1%26type%3Dpdf&ei=XaVRUabvPIaa9gSYxYHgCg&usg=AFQjCNGq-JvuhT2whqdqJWebDVSAHUOMcQ&bvm=bv.44158598,d.eWU>

Parsing the SYNTAGRUS Treebank of Russian

Joakim Nivre, Igor M. Boguslavsky, Leonid L. Iomdin

<http://www.aclweb.org/anthology-new/C/C08/C08-1081.pdf>

APPENDIX B - Consonant-Qualified Transliteration:

**An Adaptable Rule-Based System for Back-Transliteration of English
Words in Arabic Script Languages**

**Consonant-Qualified Transliteration: An Adaptable Rule-Based System
for Back-Transliteration of English Words in Arabic Script Languages**

Katherine McCreight Young

N-Space Analysis, LLC

305 Winding Trail

Xenia, OH 45385, USA

`nspaceanalysis@earthlink.net`

Abstract

We present a rule-based transliterator for back-transliteration of borrowed English words in Urdu, in which we first match the consonantal pattern against an English pronunciation dictionary and then use those possible matches as the dictionary for the vocalic matching. If the vocalic system fails to find a match, we back off to the best consonantal match. This consonant-qualified system performs better than either consonantal or full consonant-and-vowel matching, and is readily adaptable to other Arabic script languages.²⁸

1 Introduction

We are interested in transliteration as a method to translate out of vocabulary (OOV) words in the output of machine translation in a system trained using parallel Urdu and English text. When working with the parallel Urdu-English text provided as part of the Less Commonly Taught Languages (LCTL) Urdu Corpus from the Linguistic Data Consortium (LDC)²⁹ we find that two frequent types of OOV words in the output of our machine translation are named entities (NE) and borrowed English words that have been written phonetically in Urdu script. For example, the following sentence shows three OOV words that correspond to English borrowings, میڈونا /mydwnA/ *Madonna*, ٹیکنگ /tykng/ *taking*, and چانسز /čAnsZ/ *chances*.³⁰

Machine Translation

Canadian show 'ٹیکنگ چانسز' had visited میڈونا only twenty four countries while seventeen countries who were in his first visit last eight years in the world.

Reference Translation

While the Canadian show “Taking Chances” visited twenty-four countries, Madonna went to only seventeen countries as part of her first world tour in the last eight years.

Urdu writers are often opportunistic in borrowing English words, even when Urdu words are available for the same concepts. Within the LCTL Urdu corpus, the borrowed word زیرو /zyrw/ *zero* is about as common as the Urdu/Arabic word صفر /Sfr/ *zero*. We even find examples in which a word is expressed in its borrowed and native forms within the same sentence. For example:

دریں اثنا عراقی الیکشن کمیشن نے کہا ہے کہ انتخابات کے منفی نتائج آنے میں ابھی مزید دو ہفتے کا وقت لگے گا .

Reference Translation

.... there for the Iraqi election commission have announced that the positive result of election will come , after two week

In this sentence we find both the borrowed word الیکشن /Alykšn/ *election* in the phrase, *Iraqi election commission*, and later the Urdu/Arabic word انتخابات /AntxAbAt/ *elections*.

If a borrowed word occurs frequently in the parallel text, it may be successfully processed by the machine translation system. The borrowed forms of *zero* and *election* are in fact properly translated by our MT system. Less frequent borrowings, however, may remain untranslated, as in our first example. The prevalence of borrowed words therefore makes back-transliteration an important component of Urdu-English machine translation.

It is important to distinguish this back-transliteration process from forward transliteration. Names in the parallel text frequently reflect forward transliteration, in which Arabic or Urdu names have been

²⁸ [sponsorship statement]

²⁹ <http://projects.ldc.upenn.edu/LCTL>

³⁰ We use slash brackets to indicate romanization of the Urdu characters following Habash et al. (2007) and Habash and Metsky (2008).

approximated in English spelling. Transliteration of new NE is difficult, due to variations in spelling in both languages; in news articles this is complicated by the presence of names from many different languages. NE transliteration is therefore typically approached with statistical training. Aminzadeh and Shen (2008) develop a statistical system to transliterate from Urdu spellings to English spellings, using Hidden Markov Models and training on NE found in parallel text.

However, the mappings learned for NE are not necessarily appropriate for the back-transliteration of common English words. As Aminzadeh and Shen (2008) note, the system exhibits

...an effect that "Urdu-izes" borrowed words from English; for example, "airport" in Urdu gets transliterated to "arapurat," which has several hallmarks of how city names are written in Urdu.

Table 1 shows additional examples of the types of errors that can be introduced when this NE-trained system is applied to back-transliterations (we use the abbreviation A&S to indicate the Aminzadeh and Shen (2008) system).

Urdu	Romanization	English	A&S
الیکٹرانک	/Alyk t rAnk/	electronic	electrance
سسٹم	/ss tm/	system	sectem

Table 1. Back-transliteration errors using the NE-trained Aminzadeh and Shen (2008) system

We can conjecture that the system has had to learn mappings from both Urdu ک /k/ and Urdu س /s/ to English letter c, leading to some confusion in handling these borrowed common words.

While NE transliteration has to allow for wide variation in spelling, the back-transliteration of common words is more systematic. In borrowing English words, Urdu writers follow reasonably predictable patterns for consonant sounds, making it possible to develop a rule-based transliteration system for these words. We map from Urdu consonant letters to consonant sounds, and then look for these sounds in an English pronunciation dictionary. For example, an Urdu س /s/ can be reliably interpreted as the sound [s], and matched to English dictionary entries that contain [s].

Vowel mappings are more varied, but since we do not need to generate the word, but merely recognize a borrowed English word, we can use the pronunciation dictionary to select the correct vowel pattern. By checking the consonant pattern first, we reduce the search space for the vowel matching process.

We present here a rule-based Consonant-Qualified (CQ) transliterator for Arabic script languages, in which we first match the consonantal pattern against the CMU English pronunciation dictionary³¹ and then use those possible matches as the dictionary for the vocalic matching. If the vocalic system fails to find a match, we back off to the best consonantal match. We use word frequency to determine the best match among multiple choices. We find that the CQ system performs better than either full matching or consonant-only matching, and achieves a substantial improvement over the LDC-provided transliterator that is part of the LCTL Urdu Corpus, when tested on English words found in the LCTL Urdu Corpus.

The transliterator was extended to Arabic, Dari, and Pashto by additions to the letter to sound mapping. The system also permits the addition of domain-specific English dictionaries, using a letter-to-sound generator to provide the pronunciation information. In both letter mapping and dictionary creation, we achieve some utility simply by getting the consonantal patterns correct, thus reducing the amount of linguistic analysis necessary to extend the system.

2 Related Work

³¹ <http://www.speech.cs.cmu.edu/cgi-bin/cmudict>

Much of the previous work in transliteration of English words written in Arabic script involves statistically trained models of character correspondences. Stalls and Knight (1998) introduce a system for back-transliteration of Arabic words using a series of probabilistic models. As mentioned above, Aminzadeh and Shen (2008) develop a statistical transliterator for Urdu NE, using a Hidden Markov Model trained on Urdu-English NE pairs identified in parallel text. At one extreme, AbdulJaleel and Larkey (2003) describe a statistical transliteration technique for English and Arabic that "requires no heuristics or linguistic knowledge of either language."

In contrast, the LCTL Urdu transliterator (LCTL-UT) provides a rule-based model, in which Urdu consonant characters are mapped to possible English spellings. After vowelization, the resulting English character strings are matched against an English word frequency list.

We follow the rule-and-dictionary model of the LCTL-UT; however, our system differs in that we map Urdu characters to a phonemic representation, which is then matched against an English pronunciation dictionary. We also differ in our treatment of unexpressed short vowels. Urdu writers using borrowed English words may leave the short vowels [ə ɪ ɛ ʊ] unwritten, as in the word *سسٹم*/sɪstəm/ system; back-transliteration thus requires some vowelization process to restore the missing sounds. The LCTL-UT system attempts to restore missing vowels by checking every possible vowel spelling permutation between each consonant pair. Our system, in contrast, uses a rule-based vowel matching system for whatever vowels are present in the Urdu script, and a matching utility that can skip short English vowels when comparing the Urdu input to the English dictionary entries.

A key feature of our system is the separation of consonant and vowel matching. A recent statistical transliteration system for Arabic (Hermjakob et al., 2008) also separates vowel and consonant processing, using consonant skeletons to identify candidate transliterations within an English wordlist. The best transliteration is then determined based on word frequency and a transliteration cost that is derived from several hundred hand-built rules relating Arabic character substrings to English character substrings. Our system differs from Hermjakob et al. in that we match on sounds instead of English spellings, and we require an exact match during the vocalic matching phase.

3 The Transliteration System

3.1 Overview

Consonant mapping for back-transliteration is usually relatively straight-forward, while vowel mapping is complicated, involving more variation in phoneme inventory, dialect, and spelling conventions. Vowel mapping is particularly difficult with Arabic scripts, where short vowels are generally not expressed in writing. We suggest that it makes sense to distinguish between high-confidence consonant mapping and low-confidence vowel mapping when implementing a transliteration system.

In our initial work with rule-based transliteration of Urdu, we compared a consonant-only (C) mapping with a full consonant-vowel (CV) mapping.

For example, consider the word *فاسٹ* /fAsɪt/, which is a transliteration of the English word *fast*. C mapping creates the representation [f_st]. English words that match this consonantal sound pattern include {*feast*, *fist*, *faced*, *fest*, *fast*, *fussed*, ...}. In our C mapping system, we return the highest frequency entry among the matching words; in this example, this is the word *faced*.

In CV mapping, we match both vowels and consonants. Our vowel mapping specifies that the Urdu character *ا* /A/ maps to [ɪ ɛ a æ ə ɔ ʊ]. So we have to find English words that match one of these full CV patterns: {fɪst fɛst fAst fæst fəst fɔst fʊst}; this rules out *feast* [fiest] and *faced* [fæst]. The most frequent word remaining is *fast* [fæst].

In general we found that the CV transliteration was more accurate, but slower (because of the number of possible interpretations of each vowel symbol), and frequently failed to find a matching English word (because of discrepancies in how vowels were represented).

We improved performance by separating the consonant and vowel matching steps, with consonantal backoff. First, we identify the English words that satisfy the consonantal pattern; then we use this list of consonant-qualified (CQ) words as the dictionary for full CV matching. If the CV phase fails to find a match, we backoff to the best consonant-qualified match.

3.2 Details

We create a map of Urdu characters to sounds, allowing one-to-many mappings. For each word, we then construct a list of possible sound sequences, which we attempt to find in the English pronunciation dictionary.

During the consonant matching phase, we map all vowels to a placeholder, @. During the vowel matching phase, Arabic vowel characters are mapped to sets of possible vowel sounds. For example, as we noted above, Urdu ا /A/ maps to $[\text{ɪ} \text{ ɛ} \text{ a} \text{ ə} \text{ ɔ} \text{ ʊ}]$. We identify vowel mappings from reference grammars, and supplement this observationally, noting that some vowel characters are used for a wider range of sounds than traditionally indicated.

The characters و /w/ and ی /y/ require special treatment, as these can represent either vowels or consonants in Urdu: و /w/ maps to $[\text{w} \text{ v} \text{ o} \text{ ʊ} \text{ ɔ}]$, and ی /y/ maps to $[\text{y} \text{ i} \text{ e} \text{ æ} \text{ ɛ} \text{ ɪ}]$. The presence of و /w/ or ی /y/ causes the program to create variant forms for consideration (see Section 3.3).

We derive a dictionary from the 125,000-entry CMU pronunciation dictionary of American English, removing stress indications and providing one column for full vocalic representation and one column using the vowel placeholder, @. We annotate these entries with word frequency values for our training data as shown in Table 2.

Freq	English	CV	C
66	donor	downer	d@n@r
67	dinner	dihner	d@n@r
140	benefits	behnahfihts	b@n@f@ts
337	project	praajhehkt	pr@jh@kt

Table 2. Example dictionary entries³²

During the initial C matching phase, the program looks up words that match the consonantal patterns for any member of the list of sound permutations. If an Urdu vowel is present, it must match a vowel placeholder in the English entry; English vowels may be skipped when there is no corresponding Urdu vowel.

The list of consonantal matches provides the dictionary for the CV phase, in which both vowels and consonants must match, with the exception of the lax vowels $[\text{ə} \text{ ɪ} \text{ ɛ} \text{ ʊ}]$. We allow the system to skip English lax vowels, since these are often omitted in Urdu writing. When multiple matches are found, the

³² The CV entries are given in the CMU phonetic spelling, which differs from the International Phonetic Alphabet (IPA):

CMU	IPA	CMU	IPA	CMU	IPA
ow	o	ao	ɔ	aa	a
uw	u	uh	ʊ	ah	ə
iy	i	ih	ɪ	er	ɪ
ey	e	eh	ɛ	jh	dʒ

program selects the highest frequency word. If no CV match is found, the program reverts to the best C match.

3.3 An Example

We illustrate this process with the word ڈونر /dwnr/, a transliteration of the English word *donor*. We first look up the consonantal character mappings, while mapping any vowels to the placeholder, @. These mappings are shown in Table 3.

Urdu	IPA	Notes
ڈ	d	really retroflex ɖ
و	w v @	vowel place holder
ن	n	
ر	r	

Table 3. Consonant mappings for ڈونر

Urdu writers frequently use the retroflex characters ڈ ڈ /d t r/ [ɖ t ɽ] to represent English sounds, while alveolar د ر ت /d t r/ [d t ɹ] are more common for Urdu words. We map both alveolar [d] د and retroflex [ɖ] ڈ to IPA [d] for the purposes of transliteration.

Taken together, the mappings in Table 3 create the set of possible sound sequences, {dwnr, dvnr, d@nr}. We look in the CMU-based pronunciation dictionary for entries that match any of these possibilities. There are no matches for [dwnr] or [dvnr], but [d@nr] does generate matches. Since not all English vowels are spelled out when borrowed into Urdu, our matching program allows skipping of the vowel marker @. The target [d@nr] can therefore be matched in dictionary entries to the sequences [d@nr] or [d@n@r] or [d@n@r@], generating the word matches shown in Table 4.

Freq	English	CV	C
67	dinner	dihner	d@n@r
66	donor	downer	d@n@r
6	downer	dawner	d@n@r
2	dinar	dihnaar	d@n@r
0	daner	deyner	d@n@r
0	danner	daener	d@n@r
0	deaner	diyner	d@n@r
0	deener	diyner	d@n@r
0	dehner	dehner	d@n@r
0	denner	dehner	d@n@r
0	diener	diyner	d@n@r
0	diner	dayner	d@n@r
0	dohner	downer	d@n@r
0	donar	daaner	d@n@r
0	doner	daoner	d@n@r
0	donner	daaner	d@n@r
0	dooner	duwner	d@n@r
0	dyneer	dihnihr	d@n@r
0	dyneer	daynihr	d@n@r
0	denarii	dihnaeriy	d@n@r@
0	denaro	dihnaarow	d@n@r@
0	deniro	dihnihro	d@n@r@

Table 4. Consonant-qualified matches for ڈونر

The preferred match at this stage is *dinner*, because it has the highest word frequency in our English parallel data. However, we retain all the potential matches as information for the CV matching phase.

Proceeding to full CV mapping, we find the correspondences listed in Table 5. Since we are considering vocalic information, we now also include the possible syllabic use of the ر /r/ character.

Urdu	IPA	CMU
ڈ	d	d
و	w v o u ɔ ʊ	w v ow uw ao uh
ن	n	n
ر	r ɾ	r er

Table 5. Consonant-vowel mappings for ڈونر

The CV mappings generate the set of possible sound sequences shown in Table 6. We compare these possible sound sequences with the consonant-qualified list in Table 5, and find that the possible CV matches are the word *donor* and the proper names, *dohner* and *dooner*, as indicated in Table 7.

IPA	CMU	IPA	CMU
dwnr	dwnr	dwnɾ	dwner
dvnr	dvnr	dvnr	dvner
donr	downr	donɾ	downer
dunr	duwnr	dunɾ	duwner
dɔnr	daonr	dɔnr	daoner
dɒnr	duhnr	dɒnr	duhner

Table 6. CV sound sequences for ڈونر

Freq	English	CV	C
66	donor	downer	d@n@r
0	dohner	downer	d@n@r
0	dooner	duwner	d@n@r

Table 7. CV matches for ڈونر

The high frequency word *dinner* is now excluded, since the vowels do not match. We find instead the highest frequency match among the CV matches in Table 7, which is the word *donor*.

3.4 Separate Treatment of Digraphs

While the program primarily relies on a table of letter-to-sound correspondences, digraphs must be handled separately. For example, diphthongs in Urdu are generally represented by digraphs or trigraphs, such as او /Aw/ IPA [au], CMU [aw]. In order to recognize these, an early step in the program checks for digraphs and adds the diphthong interpretation as a variant sound entry for that word. At this stage we also address the Urdu convention of prepending the vowel character ا /A/ to initial clusters of [s] + consonant. When we encounter the initial digraph اس /As/, we record sound sequences for the word both with and without initial [a]. For example, when attempting to transliterate the word اسکولز /Askwɪz/ *schools* we look for both [askwɪz] and [skwɪz].

3.5 Advantages of Consonantal Backoff

By using CV matching with C backoff, we make the most of both vocalic and consonantal information. For short words like ڈونر /dwnr/ *donor*, discussed above, the consonantal match returns too many options; looking at the vowel quality allows us to select the correct match.

For longer words like بینیفٹس /bynyftɪs/ *benefits*, the consonantal pattern alone may be sufficient to identify the word. Unfortunately, the vowel pattern here does not match: ی /y/ (seen here in medial form ۄ) maps to [y i e æ ɛ ɪ], but not the needed [ə] in the second syllable (see dictionary entry in Table 2). Using a

strict CV matching program, *benefits* would remain unknown. Consonantal backoff allows us to retain this word.

Consonantal backoff also helps mitigate failures in vowel matching that result from dialectal variation. For example, our data contain the word *پروجیکٹ* /prwɨjkt/ *project*, following British English pronunciation with [o] in the first syllable. This does not match the American English pronunciation with [a] given in the CMU dictionary (see Table 2), so vowel matching fails. Our program correctly detects this word by backing off to the consonantal match. While systematic dialectal variation can be handled by adding dictionary resources (see Section 5.1), consonantal backoff provides support when such information is not available.

4 Results

We test our system against a list of 282 Urdu words identified by a native Urdu speaker as transliterations of English words. The wordlist was derived from a list of the most frequent out of vocabulary words in our processing of the LCTL Urdu corpus, and contains primarily common nouns. Words used during the development of the sound mapping rules were excluded from the test set. Arabic script forms were normalized to their Urdu variants.

When we look at the percent of correct transliterations among words attempted (Table 8), we find that CV transliteration is the most accurate, with 90.5%. However, because CV returns no result for 51 of the 282 words, the overall accuracy is only 74.1%, lower than that for the C transliterator (see Table 9). The CQ combination (vocalic matching with consonantal backoff) yields the best overall accuracy, 84.8% (Table 9).

Character error rates (CER) computed with SCLITE³³ show a similar pattern, with the CQ combination again the most effective overall. The CV method has the lowest CER among words attempted, but yields the highest overall CER, because of the number of omitted words.

	C	CV	CQ
% correct	79.4	90.5	87.9
CER	8.9	3.5	5.1
Words Attempted	272	231	272

Table 8. Transliteration performance not counting unknowns on the 282-word common word test set.

	C	CV	CQ
%correct	76.6	74.1	84.8
CER	13.2	24.5	9.5

Table 9. Overall transliteration performance on the 282-word common word test set.

The overall CQ word accuracy score of 84.8% correct on the common noun test set is well above that of the LCTL Urdu transliterator, which transliterates this test set with 18.4% accuracy. We also compare our system to the statistical Urdu transliterator of Aminzadeh and Shen (2008), henceforth, A&S. As discussed in the introduction, the A&S system was trained primarily on NE, and performs less well on common nouns, achieving a word accuracy rate of only 7.4% on this test set. Table 10 summarizes these results.

	CQ	LCTL	A&S
% correct	84.8	18.4	7.4
CER	9.5	38.8	37.3

Table 10. Comparative word accuracy and character error rates on the 282-word common-word test set.

³³ <http://www.itl.nist.gov/iad/mig/tools/>

For comparison, we considered the A&S test set, which contains both NE and borrowed English words, with an emphasis on NE (approximately 2/3 of the 191 entries are Urdu names). This test set, like our common noun test set, was drawn from the LCTL Urdu corpus by identifying transliterations in the parallel Urdu and English text. There are no overlaps in the two lists.

A&S report a CER of 35.1% on the on the 191 word A&S test set (with an improvement to 30.4% CER by pruning during optimization). Their word accuracy rate is 10.4%.

The CQ transliterator achieves a higher word accuracy of 30.9% correct on the A&S test set. In recognizing NE, the CQ transliterator is limited to the NE that are found in the CMU dictionary. Examination of the output shows that the CQ transliterator often fails to make a guess for NE, or incorrectly associates the NE with a common word. For example, the name ایهود /Ayhwd/ *ehud* is transliterated as *ahead*. Poor performance with the NE is offset, however, by greater success with the English common nouns.

The CQ transliterator returns a poor CER when compared to the A&S system, with CER of 51%. The fact that the CQ transliterator returns no guess when no dictionary match can be found leads to this higher CER.

We also ran the rule-based LCTL Urdu transliterator on this data. The LCTL system has a word accuracy rate of 22% correct, and a CER of 31.5%, reflecting the fact that this system often makes partial matches. These results are summarized in Table 11.

	CQ	LCTL	A&S
% correct	30.9	22.0	10.4
CER	51.0	31.5	35.1

Table 11. Overall word accuracy and character error rates for the A&S test set (191 words).

5 Extension to New Domains

5.1 Additional Vocabulary

The use of a pronunciation dictionary limits the program to things we have seen before. Novel words such as یوٹیوب /ywtɪwb/ *YouTube* are not recognized. In order to handle both dialectal variation and domain-specific words, we provide a capability to specify additional dictionaries. For Urdu, it will be important to add a dictionary that contains British English pronunciation. In addition, specific OOV words can be added automatically by creating a list of English words and generating their pronunciation via a letter-to-sound generator.

5.2 Additional Languages

The use of rule-based letter-to-sound mappings allows us to quickly adapt the system to other Arabic script languages. We added rules for Arabic, Dari, and Pashto, and demonstrated a basic capability to identify and translate borrowed English words in these languages. Some words were recognizable from existing character mappings (Table 12); others required the addition of a few lines to the character mapping code (Table 13). For romanization of the Pashto letters in these tables we follow the American Library Association - Library of Congress guide (ALA-LC)³⁴, in which /ŋ/ and /d/ represent retroflex consonants.

Language	Word	Letters	English
Dari	یونسف	ywnsf	unicef
Dari	واکسین	wAksyn	vaccine
Pashto	جرمني	jrmny	germany
Pashto	پولند	pwlnđ	poland

³⁴<http://www.loc.gov/catdir/cpsd/romanization/pushto.pdf>

Table 12. Transliterations using existing mappings.

Language	Word	Letters	English	Mapping
Arabic	ليفربول	lyfrbwł	liverpool	ف > f,v ب > p,b
Pashto	پولند	pwlŋd	poland	ڼ > n
Pashto	پولند	pwlŋɖ	poland	ɖ > d

Table 13. Transliterations requiring new mappings.

The example words in Tables 12 and 13 come from several sources. The Dari words are transliterations identified in a parallel-text UN article³⁵, the Arabic example comes from parallel definitions in Wikipedia, and the Pashto examples come from parallel text in the online newspaper, Sada-e-Azadi³⁶.

Note that the Pashto examples display a high level of spelling variation, alternating retroflex and non-retroflex consonants to create three spellings of *poland* within a single article. Our character-to-sound mapping handles these variations by generalizing across sounds: Both alveolar and retroflex characters are mapped into the alveolar sounds [n] and [d] before we attempt to look up the English words.

The extended character mappings introduce some errors for the original language, Urdu. For example, while Urdu has separate letters for the sounds f and v, Arabic uses one letter for f/v, requiring a new mapping, ف > f,v. Using only Urdu mappings, the program correctly transliterates the Urdu spellings, هاف /hAf/ *half* and آفر /Āfr/ *offer*, but with the additional Arabic mapping for f/v, the program returns instead the more frequent words, *have* and *over*.

The CQ transliterator may be implemented with one generalized transliteration program that will accommodate various Arabic script languages, with some overgeneralization, or with separate mappings for the different languages, selected according to the nature of the input.

In addition, the program can be extended to other alphabets with a minimum of effort. We were able to add character mappings for Cyrillic and derive transliterations for English words that have been borrowed into Russian (Table 14). In this case there is no interaction with the Urdu character set; the program merely uses the Cyrillic mappings to create sound sequences, which are then checked against the CMU-derived dictionary as usual. Here, the example words are taken from Wikipedia article titles across languages, and romanization of the Cyrillic letters follows the ALA-LC.³⁷

Language	Word	Letters	English
Russian	Ливерпуль	Liverpul'	liverpool
Russian	Мексика	Meksika	mexico

Table 14. Transliterations of Cyrillic Script

We transliterate the Russian word Мексика /meksika/ via consonantal backoff, since the final vowel fails to match the English word. This again shows the value of separating C and CV matching, even when working with a language with full vowel spelling.

We note one difficulty in extending the program to additional languages: While most of the new information is contained in the letter-to-sound tables, digraphs must be handled separately, as discussed in Section 3.4. We must write a separate clause, for example, to introduce the Arabic sequence تش /tš/ representing [č].

6 Future Work

6.1 Stemming

³⁵http://www.unama-afg.org/news/_pr/_dari/UN/2007/_september/07sep16-who-unicef-peace-day-dari.pdf

³⁶ <http://www.sada-e-azadi.net/> 16 May 2009

³⁷ <http://www.loc.gov/catdir/cpso/romanization/russian.pdf>

The CQ transliterator can be improved by combining stemming and transliteration. There are many examples in our data of words which represent a transliterated English base word, together with the Urdu plural suffix *ون* [ō]. These often co-exist with transliterations of the English plural in [s] or [z]: we find *ليڈرز* /lyḍrz/ and *ليڈرون* /lyḍrwn/ both meaning *leaders*.

We have begun testing a stemming transliterator that identifies the Urdu plural suffix *ون* [ō] and adds variants in which the suffix is replaced with the sounds [s] and [z] in anticipation of the English plural form. For example, *ليڈرون* /lyḍrwn/ generates the set { *lyḍrwn*, *lyḍrz*, *lyḍrs* }; the sound matching component then matches /lyḍrz/ with the English word, *leaders*. Retaining the original form avoids errors due to over-stemming, since the original word in [ō] is still available for matching.

In a related problem, some pre-processing will be necessary to enable transliteration of split and compounded words, such as *ايسوسي ايشن* /Ayswsy Ayšn/ *association*, *جوبائيڈن* /jwbA'yḍn/ *JoeBiden*, and multi-word representations of acronyms like *بی بی سی* /by by sy/ *BBC*.

6.2 Contextual Information

We have also experimented with using a language model to help the transliterator select among multiple options. Language model probability was not relevant to the work reported here, since the test sets consist of lists of independent words. However, when transliterating OOV words in the output of machine translation, a situation in which we have Urdu words embedded in mostly English text, we can weight the word frequency of a potential transliteration against the trigram language model probability.

6.3 Partial Vocalic Matching

Currently we require exact vocalic matching for the CV phase; in future work we may allow partial vocalic matching. A threshold could be set for the degree of vocalic matching; this could also be weighted against word frequency and language model probability.

6.4 Integration with NE Transliteration

We would like to develop a combined system, in which back-transliteration of borrowed common nouns is handled through the CQ system, while NE are processed with a statistically trained transliterator, using NE tagging to determine which transliteration system to apply.

We also need to determine whether back-transliteration of borrowed words is more useful as a pre-processing or post-processing step for the overall machine translation system.

7 Conclusion

We suggest that statistically trained transliterators are more appropriate for named entities than for common nouns, and that back-transliteration of borrowed English words should instead be handled by a rule-based system with dictionary look-up.

We introduce a two-phase transliteration process that first establishes consonant sound correspondences between the source word and the English dictionary entries. Such consonant qualified (CQ) transliteration allows us to capitalize on the best information available from both consonantal and vocalic mapping, while recognizing that we can have greater confidence in consonantal sound mappings.

CQ transliteration has potential to adapt quickly to new vocabulary or new languages, by first mapping consonant correspondences and later adding vocalic information.

1 Acknowledgments

[thanks to M. Afzal Upal, Ryan Aminzadeh]

2 References

- 1** Nasreen AbdulJaleel and Leah S. Larkey. 2003. Statistical transliteration for English-Arabic cross language information retrieval. In *Proc. ACM CIKM*, pp. 139-146.
- 2** A. Ryan Aminzadeh and Wade Shen. 2008. Low-Resource Speech Translation of Urdu to English Using Semi-Supervised Part-of-Speech Tagging and Transliteration. 2008 IEEE Workshop on Spoken Language Technology, Goa, India.
- 3** Habash, Nizar and Hayden Metsky, 2008. Automatic Learning of Morphological Variations for Handling Out-of-Vocabulary Terms in Urdu-English Machine Translation. In *Proceedings of the Association for Machine Translation in the Americas (AMTA-2008)*. Waikiki, Hawaii.
- 4** Nizar Habash, Abdelhadi Soudi, and Tim Buckwalter, 2007. On Arabic Transliteration. In A. van den Bosch and A. Soudi, editors, *Arabic Computational Morphology: Knowledge-based and Empirical Methods*. Springer.
- 5** Ulf Hermjakob, Kevin Knight, and Hal Daumé III. 2008. Name translation in statistical machine translation - learning when to transliterate. In *Proc. of ACL/HLT*, pp. 389-397, Columbus, Ohio.
- 6** Bonnie Glover Stalls and Kevin Knight. 1998. Translating names and technical terms in Arabic text. In *Proc. of the COLING/ACL Workshop on Computational Approaches to Semitic Languages*, pp. 34-41.